



DUCO Core インターフェースマニユアル

1.8

中科新松有限公司

2026 年 10 月 10 日

目次

1	TCP 及 IP インターフェース	1
1.1	2000 ポート	1
1.2	2001 ポート	3
2	Modbus TCP Server	8
2.1	Modbus TCP Server	8
3	レシピインターフェース	11
3.1	システムデータマッピング	11
3.2	レシピの作成	14
3.2.1	ストリームデータレシピ	14
	ユーザーカスタムレシピ	14
	システムデータレシピ	15
3.2.2	CAN プロトコルレシピ	17
	ユーザーカスタムレシピ	17
	システムデータレシピ	19
3.3	レシピの使用	20
3.3.1	485 ポートレシピの使用	20
	カスタムレシピの使用	20
	システムレシピの使用	23
3.3.2	CAN ポートレシピの使用	24
	カスタムレシピの使用	24
	システムレシピの使用	26
3.3.3	TCP 及 UDP ポートレシピの使用	28
	udp server	28
	tcp/udp client	29
4	profinet device インターフェース操作マニュアル	31
4.1	Profinet device インターフェース設定	31
4.1.1	インターフェースの説明	31
4.1.2	スクリプト関数	31
4.1.3	設定手順	32
4.2	Slot データ構造	39
4.2.1	Slot データ構造	39
5	Ethernet IP インターフェース操作マニュアル	42
5.1	Ethernet 及 IP インターフェース設定	42
5.1.1	インターフェースの説明	42
5.1.2	スクリプト関数	42

5.1.3	設定手順	43
	ロボット（スレーブ）の設定	43
	スキャナ（マスター）の設定	47
5.2	データ構造	58
5.2.1	データ構造	58
	データ方向：ロボット → PLC	58
	データ方向：PLC → ロボット	60

第
ONE

TCP 及 IP インターフェース

ロボットは電源投入状態になると、TCP Server としてポート 2000 および 2001 で接続を待ち受けます。ポート 2000 はロボットへの制御コマンドを受信し、状態を返します。ポート 2001 は、ロボットの現在の状態情報を 10 Hz 周期で外部へ送信します。

1.1 2000 ポート

ポート 2000 は要求・応答方式を採用し、システムコマンドを受信します。Client と Server 間の通信プロトコルを以下に示します。

表 1: 2000 ポートコマンド

受信コマンド	戻り値	説明
run(プログラム名) または run(プログラム名, 速度オーバーライド)	開始時の応答: run start 失敗時の応答: run fail 完了時の応答: script finish	指定プログラムを実行します。速度オーバーライドが未指定または無効な場合は、現在の速度で実行します。
<start> スクリプト <end>	<start> 受信時の応答: prog mode start <end> 受信時の応答: prog mode end 失敗時の応答: run fail 完了時の応答: script finish 構文エラー時の応答: script error	スクリプトコマンドを送信する場合は、<start> で開始し、<end> で終了します。 ロボットアームは <start> と <end> の間にあるコマンドを実行します。スクリプト全体は複数回に分けて送信できます。
speed(速度オーバーライド)	成功時の応答: set success 失敗時の応答: set fail	プログラムの実行速度オーバーライドを変更します。範囲は (0, 100] です。変更に成功すると set success、失敗すると set fail を返します。
stop	成功時の応答: stop success 失敗時の応答: stop fail	実行中のコマンドを停止します。停止後は stop success を返します。ロボットが実行状態でない場合に stop コマンドを受信すると、stop fail を返します。

次のページに続く

表 1 – 前のページからの続き

受信コマンド	戻り値	説明
pause	成功時の応答: pause success 失敗時の応答: pause fail	実行中のコマンドを一時停止します。一時停止後は pause success を返します。ロボットが実行状態でない場合に pause コマンドを受信すると、 pause fail を返します。
resume	成功時の応答: resume success 失敗時の応答: resume fail	一時停止中のロボットは実行を再開し、 resume success を返します。一時停止状態でない場合に resume コマンドを受信すると、 resume fail を返します。
poweron	成功時の応答: poweron success 失敗時の応答: poweron fail	ロボットの電源を投入します。成功すると通信リンクを確立して poweron success 、失敗すると poweron fail を返します。すでに電源投入済みの場合は already poweron を返します。
poweroff	成功時の応答: poweroff success 失敗時の応答: poweroff fail	ロボットの電源を切ります。成功すると poweroff success 、失敗すると poweroff fail を返します。すでに電源切断状態の場合は already poweroff を返します。
enable	成功時の応答: enable success 失敗時の応答: enable fail	ロボットをイネーブルにします。成功すると enable success 、失敗すると enable fail を返します。すでにイネーブル状態の場合は already enable を返します。
disable	成功時の応答: disable success 失敗時の応答: disable fail	ロボットをディセーブルにします。成功すると disable success 、失敗すると disable fail を返します。電源投入済みで未イネーブル状態の場合は already disable を返します。
shutdown	成功時の応答: shutdown success 失敗時の応答: shutdown fail	ロボットが poweroff 状態の場合、ロボットシステムを自動終了して shutdown success を返します。ロボットの電源が切れていない場合は shutdown fail を返します。
state	各システム状態を次の形式で返します: (1):(2):(3):(4) 注: (1) ロボット状態、(2) プログラム実行状態、(3) 安全監視状態、(4) ロボット動作モード	各状態は数値で返されます。各値の意味については、ポート 2001 のアドレス 1448~1451 を参照してください。
isprogfinish	プログラムの実行終了状態を返します。終了時は 1、未終了時は 0 を返します。	

次のページに続く

表 1 – 前のページからの続き

受信コマンド	戻り値	説明
getlasterror	ロボットで最後に発生したエラーの詳細情報を返します。形式： error_id:error_message	エラーコードは 16 進数で返されます。
getsysteminfo	ロボットのシリーズ、型式、SN、ソフトウェア／ファームウェアバージョンなどの情報を返します。	

1.2 2001 ポート

ポート 2001 はロボットの現在の状態情報を 10 Hz 周期で外部へ送信します。データの全長は 1468 byte です。

表 2: 2001 ポート状態データマッピング

データ	型	個数	バイト数	アドレス	説明
関節実位置	float	7	28	0-27	6 関節の各実位置。第 7 要素は予約。単位: rad
関節実速度	float	7	28	28-55	6 関節の各実速度。第 7 要素は予約。単位: rad/s
関節実加速度	float	7	28	56-83	6 関節の各実加速度。第 7 要素は予約。単位: rad/
関節実トルク	float	7	28	84-111	6 関節の各実トルク。第 7 要素は予約。単位: Nm
関節指令位置	float	7	28	112-139	6 関節の各指令位置。第 7 要素は予約。単位: rad
関節指令速度	float	7	28	140-167	6 関節の各指令速度。第 7 要素は予約。単位: rad/s
関節指令加速度	float	7	28	168-195	6 関節の各指令加速度。第 7 要素は予約。単位: rad/
関節指令トルク	float	7	28	196-223	6 関節の各指令トルク。第 7 要素は予約。単位: Nm
関節実温度	float	7	28	224-251	7 関節の実温度。予約データ。
関節実電流	float	7	28	252-279	6 関節の各実電流。第 7 要素は予約。単位: 定格電流に対する千分率。
サーボドライブエラー ID	uint	7	28	280-307	6 関節のサーボドライブエラー ID。第 7 要素は予約。
サーボドライブステータスワード	uint	7	28	308-335	6 関節のサーボドライブステータスワード。第 7 要素は予約。
予約	byte	32	32	336-367	

次のページに続く

表 2 – 前のページからの続き

データ	型	個数	バイト数	アドレス	説明
TCP 実位置	float	6	24	368-391	TCP の直交座標空間における実位置姿勢。ベース座標系基準の値。データ定義：X、Y、Z、Rx、Ry、Rz、単位：m、rad
TCP 実速度	float	6	24	392-415	TCP の直交座標空間における実速度。ベース座標系基準の値。データ定義：X、Y、Z、Rx、Ry、Rz、単位：m、rad/s
TCP 実加速度	float	6	24	416-439	TCP の直交座標空間における実加速度。ベース座標系基準の値。データ定義：X、Y、Z、Rx、Ry、Rz、単位：m/s ² 、rad/s ²
フランジ実外力	float	6	24	440-463	ロボット先端フランジに作用する外力。先端力覚センサ設定時、この値はセンサデータで表した、ロボットの先端フランジ座標系における力です。データ定義：X、Y、Z、Rx、Ry、Rz、単位：N、Nm
TCP 指令位置	float	6	24	464-487	TCP の直交座標空間における指令位置姿勢。ベース座標系基準の値。データ定義：X、Y、Z、Rx、Ry、Rz、単位：m、rad
TCP 指令速度	float	6	24	488-511	TCP の直交座標空間における指令速度。ベース座標系基準の値。データ定義：X、Y、Z、Rx、Ry、Rz、単位：m/s、rad/s
TCP 指令加速度	float	6	24	512-535	TCP の直交座標空間における指令加速度。ベース座標系基準の値。データ定義：X、Y、Z、Rx、Ry、Rz、単位：m/s ² 、rad/s ²
フランジ理論外力	float	6	24	536-559	現在の動作状態におけるロボット先端フランジの理論外力です。データ定義：X、Y、Z、Rx、Ry、Rz。単位：N、Nm
ベース実外力	float	6	24	560-583	ベースに実際に作用する力をベース座標系で表します。ベース力覚センサを設定した場合、この値は力覚センサの測定値です。データ定義：X、Y、Z、Rx、Ry、Rz。単位：N、Nm

次のページに続く

表 2 – 前のページからの続き

データ	型	個数	バイト数	アドレス	説明
ベース理論外力	float	6	24	584-607	現在の動作状態におけるベースの理論作用力をロボットベース座標系で表します。データ定義：X、Y、Z、Rx、Ry、Rz。単位：N，Nm
現在有効なツール座標系	float	6	24	608-631	データ定義：X，Y，Z，Rx，Ry，Rz，単位：m，rad
現在有効なワーク座標系	float	6	24	632-655	現在のツール座標系オフセットは含みません。データ定義：X、Y、Z、Rx、Ry、Rz。単位：m，rad
合成並進速度	float	1	4	656-659	TCP 合成並進速度。単位：m/s
グローバル速度	byte	1	1	660	グローバル速度オーバーライド
Jog 速度	byte	1	1	661	Jog 速度オーバーライド
予約	byte	58	58	662-719	
機能デジタル I/O 入力	byte	8	8	720-727	1 byte で 1 つの bool 値を表します。True=1, false=0。対応ハードウェア：FDI1~FDI8
機能デジタル I/O 出力	byte	8	8	728-735	1 byte で 1 つの bool 値を表します。True=1, false=0。対応ハードウェア：FD01~FD08
デジタル I/O 入力	byte	16	16	736-751	汎用 DI 入力 1~16 に対応。対応ハードウェア：DI1~DI16
デジタル I/O 出力	byte	16	16	752-767	汎用 DO 出力 1~16。対応ハードウェア：DO1~DO16
アナログ入力	float	8	32	768-799	アナログ入力 8 点。前 4 点は電流、後 4 点は電圧。対応ハードウェア：AI_C1~AI_C4、AI_V1~AI_V4。値の範囲はハードウェアマニュアルを参照してください。
アナログ出力	float	8	32	800-831	予約
float レジスタ入力	float	32	128	832-959	Modbus、PROFINET などの外部インターフェースから変更可能
float レジスタ出力	float	32	128	960-1087	スクリプト関数から変更可能
機能 bool レジスタ入力	byte	16	16	1088-1103	Modbus、PROFINET などの外部インターフェースから変更可能
機能 bool レジスタ出力	byte	16	16	1104-1119	スクリプト関数から変更可能
bool レジスタ入力	byte	64	64	1120-1183	Modbus、PROFINET などの外部インターフェースから変更可能

次のページに続く

表 2 – 前のページからの続き

データ	型	個数	バイト数	アドレス	説明
bool レジスタ 出力	byte	64	64	1184– 1247	スクリプト関数から変更可能
word レジスタ 入力	char	32	64	1248– 1311	Modbus、PROFINET などの外部 インターフェースから変更可 能
word レジスタ 出力	char	32	64	1312– 1375	スクリプト関数から変更可能
予約	byte	32	31	1376– 1406	
シミュレーシ ョン／実機	byte	1	1	1407	ロボットシミュレーション／ 実機モード
ツール I/O 入 力	byte	8	8	1408– 1415	ツールデジタル入力 (1～2：先 端入力、3～8：予約)。対応ハ ードウェア：ツール I/O デジ タル入力 1～2
ツール I/O 出 力	byte	8	8	1416– 1423	ツールデジタル出力 (1～2：先 端出力、3～8：予約)。対応ハ ードウェア：ツール I/O デジ タル出力 1～2
ツールアナロ グ入力	float	2	16	1424– 1431	ツール先端の電圧アナログ入 力。範囲：0～10 V
ツールアナロ グ出力	float	2	16	1432– 1439	予約
ツールボタン 状態	byte	2	2	1440– 1441	1 byte で 1 つの bool 値を表 します。True=1, false=0。アド レス 1440 は S キー、1441 は T キーの状態を表します。
予約	byte	6	6	1442– 1447	
ロボット動作 モード	char	1	1	1448	0：kManual 1：kAuto 2：kRemote
ロボット状態	char	1	1	1449	0：SR_Start 1：SR_Initialize 2：SR_Logout 3：SR_Login 4：SR_PowerOff 5：SR_Disable/SR_PowerOn (電源投入・未イネーブル状態) 6：SR_Enable

次のページに続く

表 2 – 前のページからの続き

データ	型	個数	バイト数	アドレス	説明
プログラム実行状態	char	1	1	1450	0 : SP_Stopped 1 : SP_Stopping 2 : SP_Running 3 : SP_Paused 4 : SP_Pausing 5 : SP_TaskRunning
安全監視状態	char	1	1	1451	0 : SS_INIT 2 : SS_WAIT 3 : SS_CONFIG 4 : SS_POWER_OFF 5 : SS_RUN 6 : SS_RECOVERY 7 : SS_STOP2 8 : SS_STOP1 9 : SS_STOP0 10 : SS_MODEL 12 : SS_REDUCE 13 : SS_BOOT 14 : SS_FAIL 15 : SS_UPDATE
衝突検出トリガ信号	char	1	1	1452	1 は衝突検出の作動を表します。
衝突軸	char	1	1	1453	戻り値は数値です。1～6 は衝突が発生した関節を表します。11～16 はベース力覚センサに基づく衝突方向を表します。順序は X、Y、Z、Rx、Ry、Rz です。20 はエンド力覚センサに基づく衝突発生を表します。
予約	byte	2	2	1454– 1455	
ロボットエラーコード	uint	1	4	1456– 1459	最新エラーコード。複数エラー同時発生時は、いずれか 1 つのエラーコードが返される場合があります。
予約	byte	8	8	1460– 1467	

第
TWO

MODBUS TCP SERVER

2.1 Modbus TCP Server

本章では、ロボット側を Modbus/TCP server として使用する場合に送受信できるデータについて説明します。コイル (Coils)：読み出し (ファンクションコード：1)、書き込み (ファンクションコード：5 = 単一 / 15 = 複数)

表 1: コイル (Coils) アドレスマッピング

Addr	項目説明	数量	読み書き権限	機能
0-15	機能 bool レジスタ入力	16	R/W	外部からレジスタを変更し、機能をトリガーします。
16-31	機能 bool レジスタ出力	16	R	レジスタ値の変化によって、対応するシステム状態を示します。
32-95	bool レジスタ入力	64	R/W	外部からレジスタを変更し、スクリプト内で読み出します。
96-159	bool レジスタ出力	64	R	スクリプトまたは UI でレジスタを変更し、外部へ出力します。
160-167	機能デジタル I/O 入力	8	R	外部物理 I/O 入力によって機能をトリガーします。
168-175	機能デジタル I/O 出力	8	R	物理 I/O 出力によって、対応するシステム状態を示します。
176-191	デジタル I/O 入力	16	R	外部物理 I/O 入力を、スクリプト、UI、または外部ポートから読み出します。
192-207	デジタル I/O 出力	16	R/W	スクリプト、UI、または外部ポートから I/O を変更し、外部物理 I/O へ出力します。
208-215	ツール I/O 入力	8	R	外部物理 I/O 入力を、スクリプト、UI、または外部ポートから読み出します。
216-223	ツール I/O 出力	8	R/W	スクリプト、UI、または外部ポートから I/O を変更し、外部物理 I/O へ出力します。
224-231	ツール先端ボタン	8	R	ツール先端ボタンの入力によって機能をトリガーします。

Holding Registers：読み出し (ファンクションコード：3)、書き込み (ファンクションコ

ード：6 = 単一 / 16 = 複数)

表 2: Holding Registers アドレスマッピング

Addr	項目説明	読み書き権限
0	1 の位：ロボット動作モード 0：kManual 1：kAuto 2：kRemote 100 の位：シミュレーション／仮想モード 0：実機モード 1：シミュレーションモード	R
1	ロボット状態 0：SR_Start 1：SR_Initialize 2：SR_Logout 3：SR_Login 4：SR_PowerOff 5：SR_Disable/SR_PowerOn（電源投入済み、未イネーブル状態） 6：SR_Enable	R
2	プログラム状態 0：SP_Stopped 1：SP_Stopping 2：SP_Running 3：SP_Paused 4：SP_Pausing 5：SP_TaskRunning	R

次のページに続く

表 2 – 前のページからの続き

Addr	項目説明	読み書き権限
3	ロボット安全状態 0 : SS_INIT 2 : SS_WAIT 3 : SS_CONFIG 4 : SS_POWER_OFF 5 : SS_RUN 6 : SS_RECOVERY 7 : SS_STOP2 8 : SS_STOP1 9 : SS_STOP0 10 : SS_MODEL 12 : SS_REDUCE 13 : SS_BOOT 14 : SS_FAIL 15 : SS_UPDATE	R
4-6	予約	R
7	TCP 合成並進速度	R
8-39	Word 入力レジスタ	R/W
40-71	Word 出力レジスタ	R
72-79	制御盤アナログ入力。count 値で表し、実電流値は count/100、 電圧値は count/1000 です。	R
80-81	制御盤アナログ I/O 電流出力。count 値で表し、実電流値は count/100 です。	R/W
82-83	予約	R/W
84-85	制御盤アナログ電圧出力。count 値で表し、実電圧値は count/ 1000 です。	R/W
86-87	予約	R/W
88-89	ツール先端アナログ入力。count 値で表し、実電流値は count/ 100、電圧値は count/1000 です。	R
90	グローバル速度オーバーライド	R/W
91	Jog 速度オーバーライド	R
92-98	関節実位置。値域：-32767～32767、単位：1/1000 rad。	R
99-105	関節実速度。値域：-32767～32767、単位：1/1000 rad/s。	R
106-111	直交実位置。値域：-32767～32767、単位：1/10000 m、1/10000 rad。	R
112-117	直交実速度。値域：-32767～32767、単位：1/10000 (m/s)、 1/10000 (rad/s)。	R

第 THREE

レシピインターフェース

レシピの分類：

(1) 機能により、ユーザーカスタムレシピ（非リアルタイムレシピ）とシステムデータレシピ（リアルタイムレシピ）に分類されます。カスタムレシピ：送受信するデータ形式をユーザーが定義します。レシピマネージャは入力データから有効データを抽出し、出力データをフレーム化します。送受信処理は、いずれもスクリプトプログラムで制御します。システムデータレシピ：ユーザーは、システムが提供するシステムデータから必要な項目を選択し、出力側からリアルタイムで読み出すか、入力側からロボットのシステム変数をリアルタイムで変更します。設定可能な変数はすべてシステムから提供され、ユーザーによる独自定義はできません。データの送受信はシステムがリアルタイムで制御し、ユーザーは制御できません。制御周期（送受信の最短周期）は **4 ms** です。

(2) プロトコルにより、ストリームプロトコルレシピと **CAN** プロトコルレシピに分類されます。ストリームプロトコルレシピは、ストリームデータ（RS-485、TCP など）に使用します。

CAN プロトコルレシピは **CAN** のみをサポートします。

3.1 システムデータマッピング

レシピのデータ型：

- (1) byte(符号なし 8 ビット整数)
- (2) char(符号付き 8 ビット整数)
- (3) word(符号なし 16 ビット整数)
- (4) short(符号付き 16 ビット整数)
- (5) dword(符号なし 32 ビット整数)
- (6) int(符号付き 32 ビット整数)
- (7) float(32 ビット浮動小数点数)
- (8) double(64 ビット倍精度浮動小数点数)

システムデータ入力：

データ方向：ユーザー ☐ ロボット

レシビ変数	型	対応インターフェース	説明
DigitalOutputCommand[1..16]	byte	制御盤汎用 D0 D0[1..16]	ユーザーがロボットの D0 を制御。 1 byte が順に 8 個の D0。
DigitalOutputCommand[17..32]	byte	ロボットアーム先端ツール出力 Tool_D0[1..2]	byte 先頭 2 bit がツール D01、D02。残り 6 bit は予約。
BitInputReg[1..2]	byte	機能レジスタ入力 fun_reg_in[1..16]	1 byte が順に 8 個の機能レジスタ入力
BitInputReg[3..10]	byte	レジスタ Bool 入力 bool_reg_in[1..64]	1 byte が 8 個のレジスタに順番に対応する bool 入力
WordInputReg[1..32]	word	レジスタ Word 入力 word_reg_in[1..32]	レジスタに順番に対応 Word 入力 1~32
FloatInputReg[1..32]	float	レジスタ Float 入力 float_reg_in[1..32]	レジスタに順番に対応 Float 入力 1~32
ForceEnd[1..6]	float	先端力覚センサデータ 制御盤汎用 D0	先端力覚センサデータをロボット 先端 TCP 座標系の力で表す。データ 定義：X、Y、Z、Rx、Ry、Rz。単位 ：N、Nm
ForceBase[1..6]	float	ベース力覚センサデータ	ベース力覚センサデータをロボット ベース座標系の力で表す。データ定 義：X、Y、Z、Rx、Ry、Rz。単位： N、Nm
RobotInfo[1..16]	float	TCP 合成並進速度など	グローバル速度比率、Jog 速度比率 、TCP 合成並進速度、エラー ID を含む

システムデータ出力：

データ方向：ロボット ☐ ユーザー

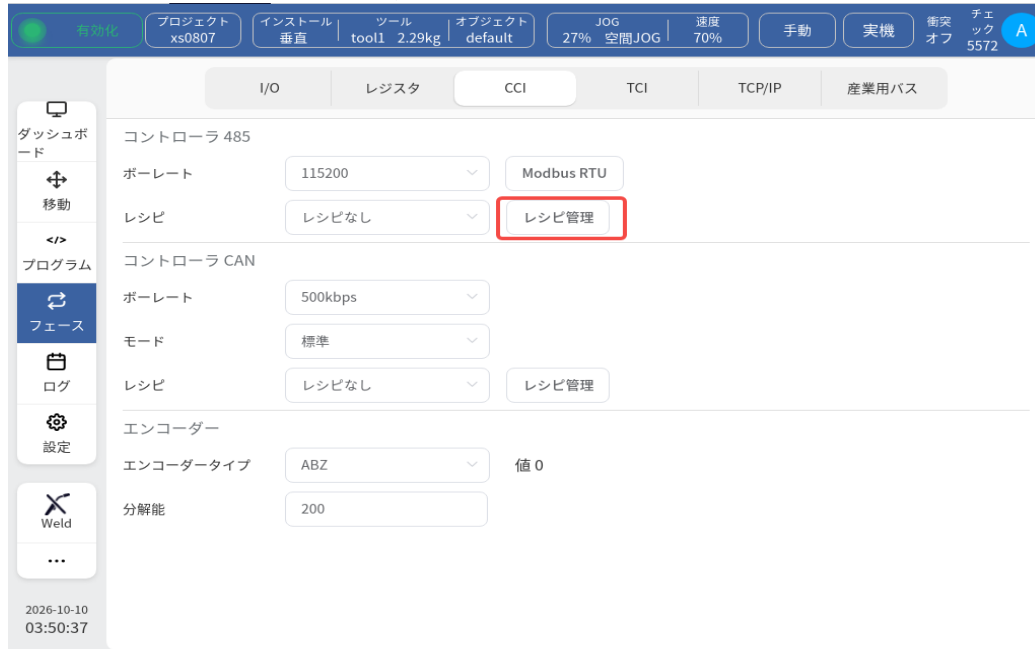
レシピ変数	型	対応インターフェース	説明
DigitalSignalStatus[	byte	制御盤汎用入力 DI[1..16]	ロボットからユーザーへ現在の 入力信号を返す。1 byte が順に 8 点の DI に対応
DigitalSignalStatus[	byte	ロボットアーム先端ツール 入力 Tool_DI[1..2]	byte 先頭 2 bit がツール DI1、DI2。残り 6 bit は予約
DigitalSignalStatus[	byte	制御盤汎用 DO DO[1..16]	ロボットからユーザーへ現在の 出力信号を返す。1 byte が順に 8 点の DO に対応
DigitalSignalStatus[	byte	ツール出力 Tool_DO [1..2]	byte 先頭 2 bit がツール DO1、DO2。残り 6 bit は予約
DigitalSignalStatus[	byte	I/O 機能入力 fun_io_in[1..8]	1 byte が機能入出力の 入力 1～8 に順番に対応
DigitalSignalStatus[	byte	I/O 機能出力 fun_io_out [1..8]	1 byte が機能入出力の 出力 1～8 に順番に対応
DigitalSignalStatus[	byte	安全入力状態	各 bit が 1 点の安全入力に対 応
DigitalSignalStatus[	byte	安全出力状態	各 bit が 1 点の安全出力に対 応
RobotState[1..7]	float	関節実位置	6 関節の各実位置。第 第 7 要素は予約。単位：rad
RobotState[8..13]	float	ベース座標系における TCP 実直交位置姿勢	TCP の実直交座標空間における 位置姿 勢。ベース座標系基準の TCP 値。 データ定義：X、Y、Z、Rx、Ry、Rz。 単位：m、rad
RobotState[14..19]	float	直交実力・トルク	先端力覚センサデータをロボッ ト 先端フランジ座標系の力。デー データ定義：X、Y、Z、Rx、Ry、Rz。 単位 ：N、Nm
RobotState[20..25]	float	ツール座標系オフセット	ツール座標系の位置姿勢 オフセット
RobotState[26..29]	float	ペイロード質量と重心	先端ペイロードの質量と重心。 単位：kg、m
RobotState[30..32]	float	予約	なし
BitOutputReg[1..2]	byte	機能レジスタ出力 fun_reg_out[1..16]	1 byte が 8 個のレジスタに 順番に対応する機能出力
BitOutputReg[3..10]	byte	レジスタ Bool 出力 bool_reg_out[1..64]	1 byte が 8 個のレジスタに 順番に対応する bool 出力
WordOutputReg[1..32]	word	Word レジスタ出力 word_reg_out[1..32]	Word レジスタに順番に対応 出力 1～32
FloatOutputReg[1..32]	float	レジスタ Float 出力 float_reg_out[1..32]	Float レジスタに順番に対応 出力 1～32
RobotInfo[1..16]	float	TCP 合成並進速度など	速度比率、TCP 合成並進速度、 エラー ID

3.2 レシピの作成

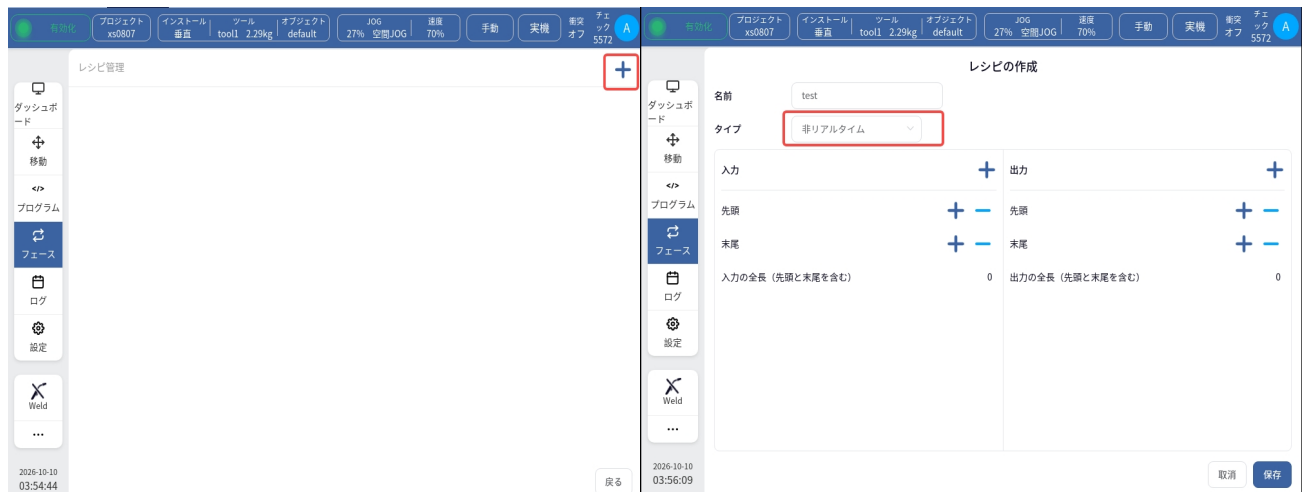
3.2.1 ストリームデータレシピ

ユーザーカスタムレシピ

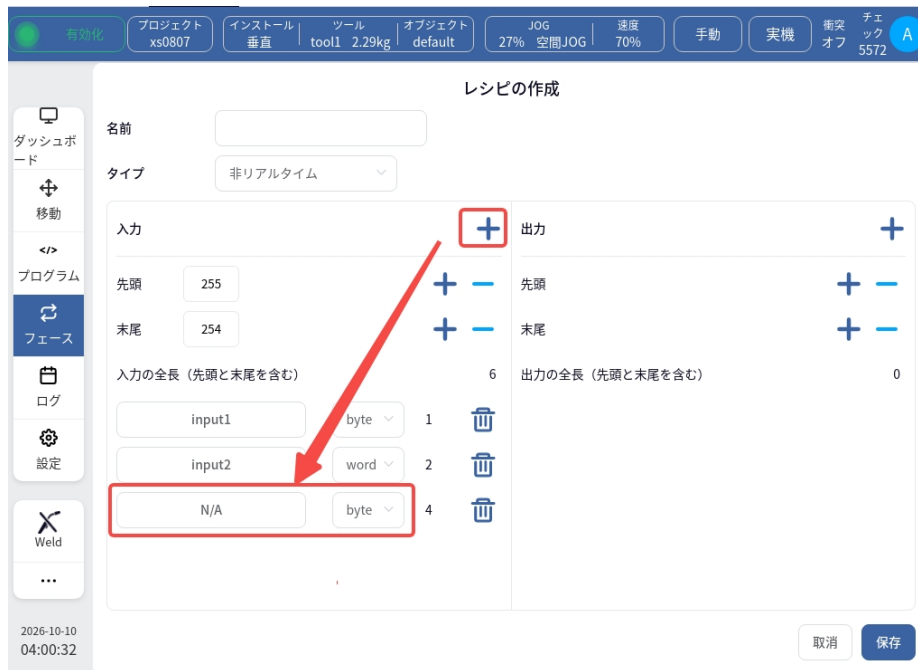
1. UI 画面で任意のストリームポート (RS-485、TCP など) のレシピ管理を選択します (ここでは 485 ポートを選択)。



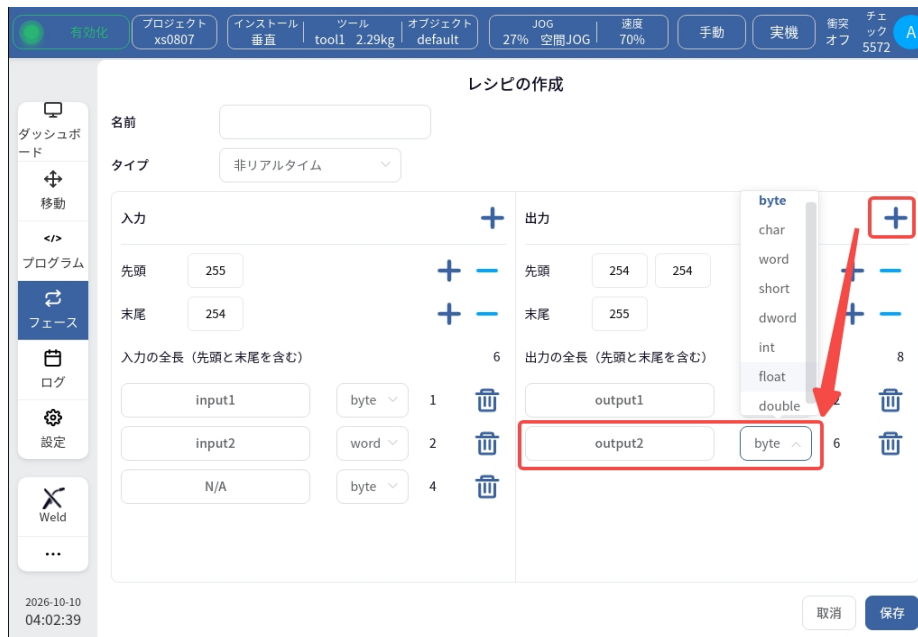
2. 「+」をクリックし、レシピ名を入力して、非リアルタイムレシピを選択します。



3. 入力データ項目を追加します。



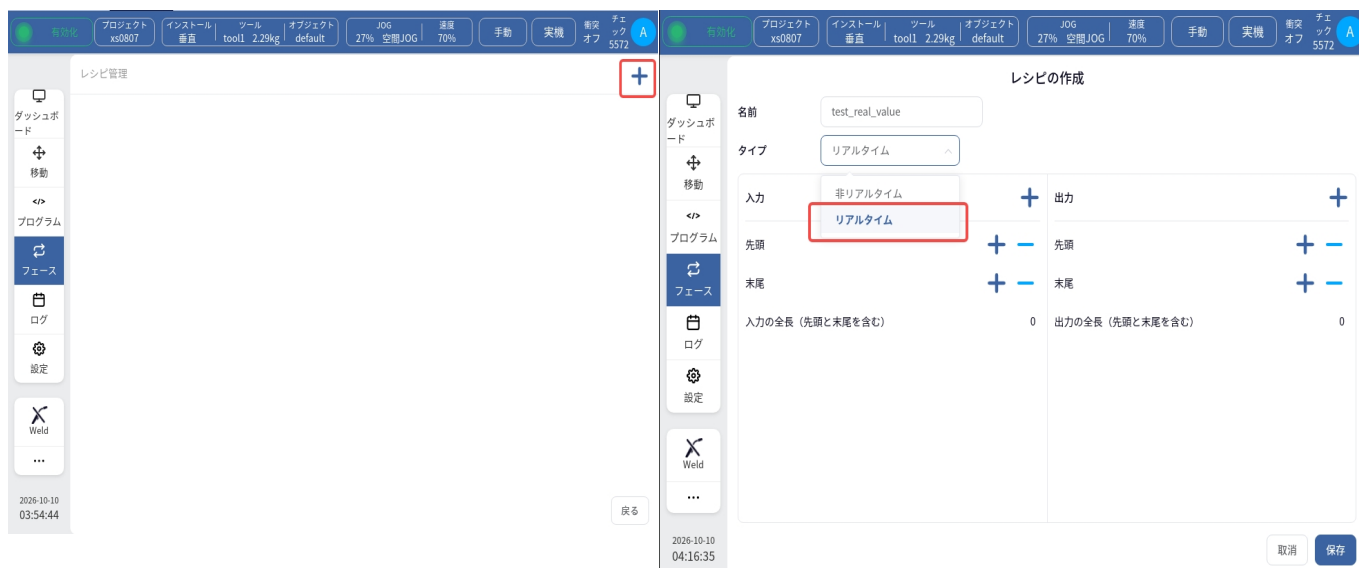
4. 出力データ項目を追加します。



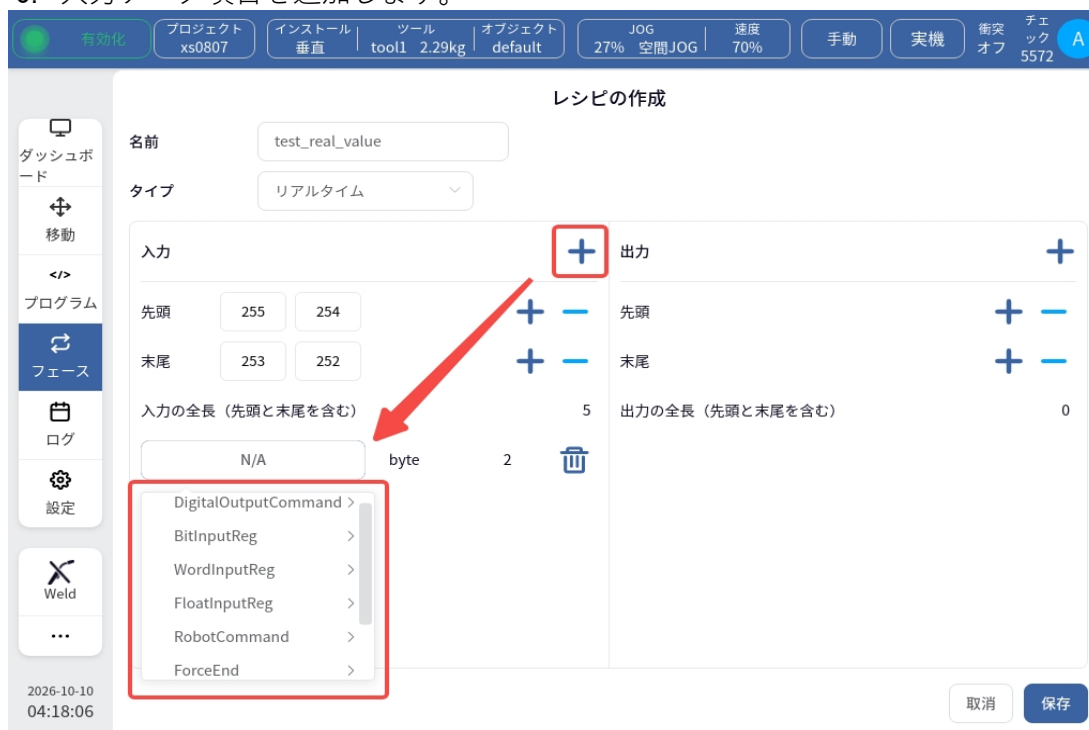
「保存」をクリックすると、レシピの作成が完了します。入力と出力のフレームヘッダおよびフレーム末尾は互いに独立しているため、それぞれ個別に設定してください。

システムデータレシピ

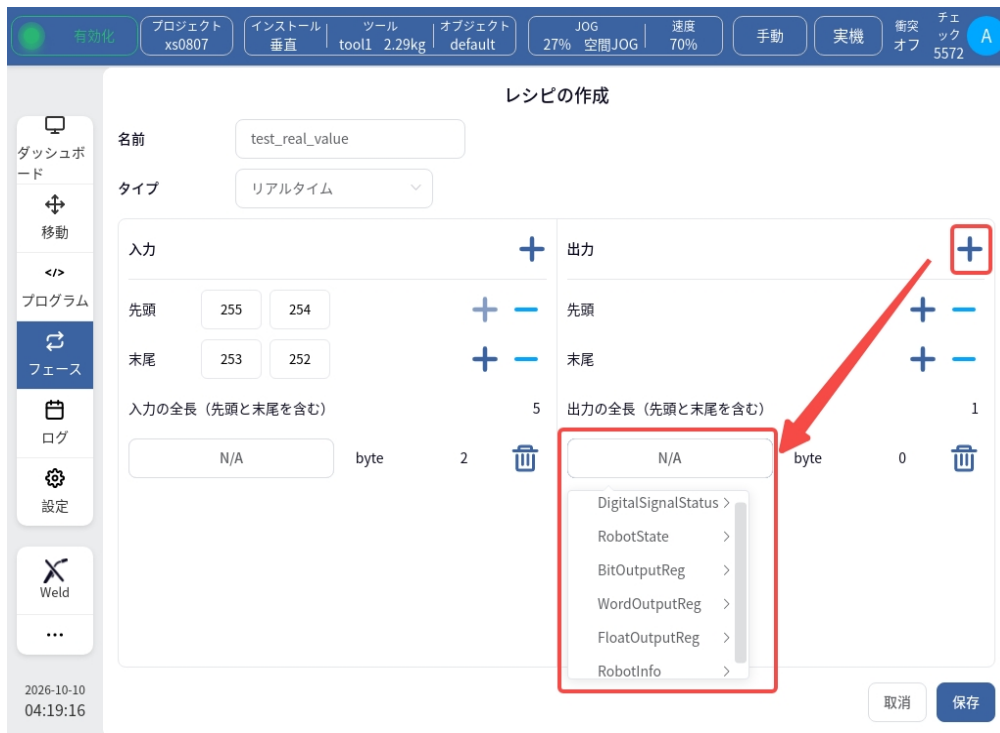
1. UI 画面で任意のストリームポート (RS-485、TCP など) のレシピ管理を選択します (ここでは 485 ポートを選択)。
2. 「**+**」をクリックし、レシピ名を入力して、非リアルタイムレシピを選択します。



3. 入力データ項目を追加します。



4. 出力データ項目を追加します。



5. 「保存」をクリックすると、レシピの作成が完了します。

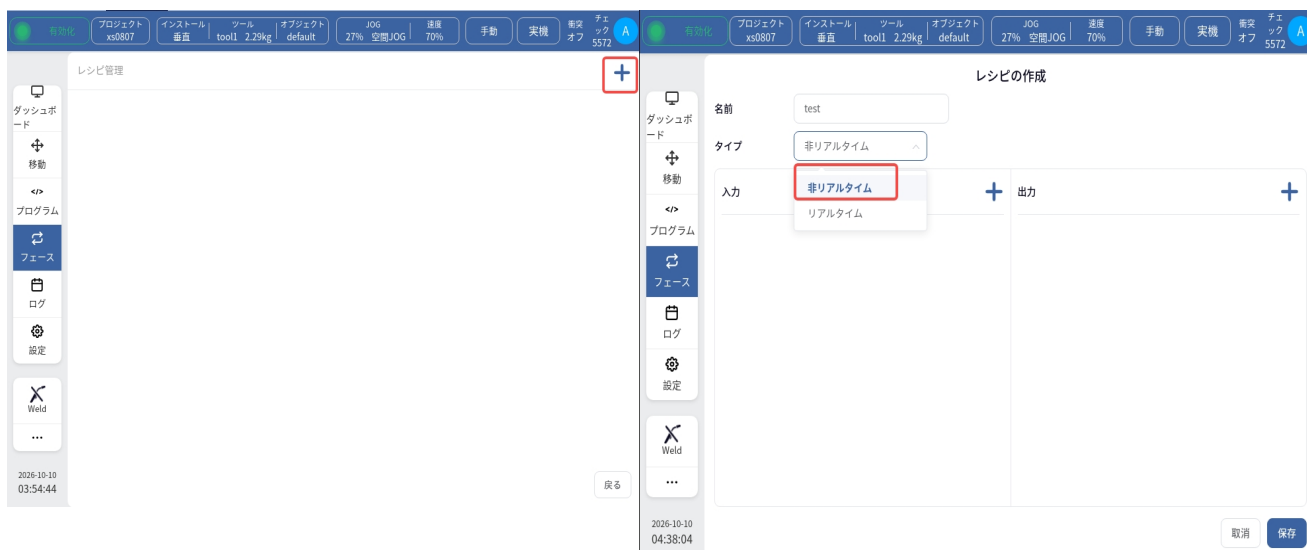
3.2.2 CAN プロトコルレシピ

ユーザーカスタムレシピ

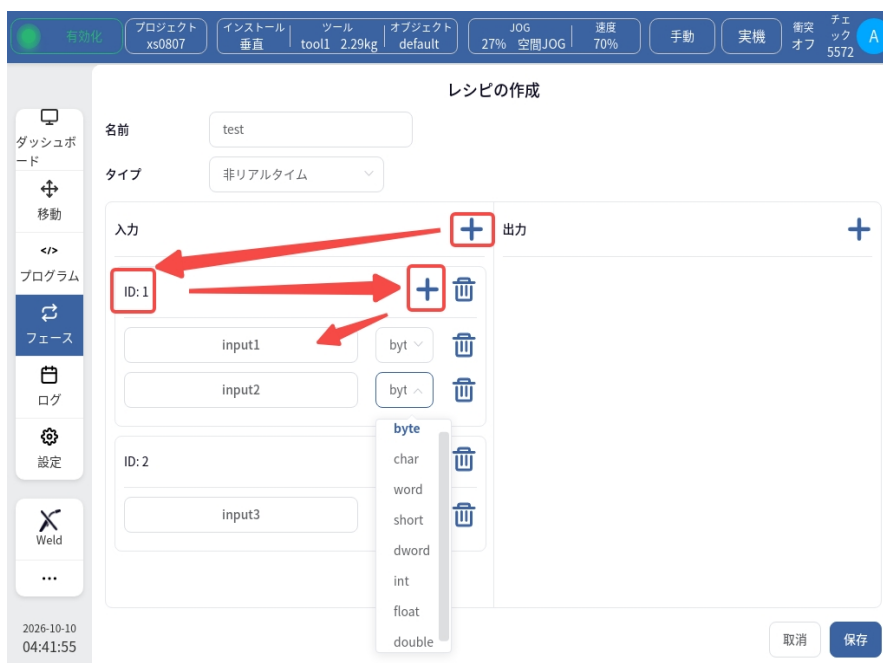
1. UI 画面で CAN ポートのレシピ管理を選択します。



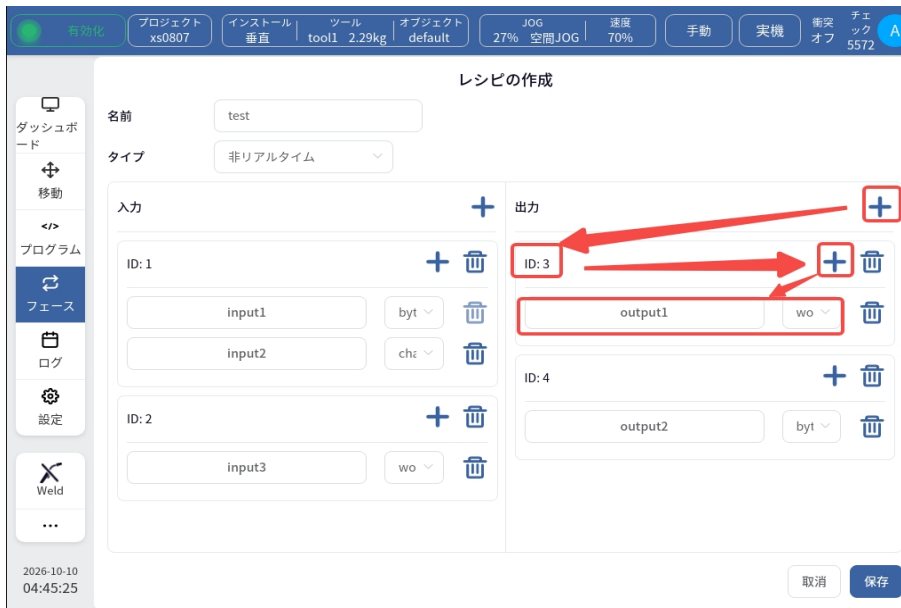
2. 「+」をクリックし、レシピ名を入力して、非リアルタイムレシピを選択します。



3. 入力データ項目を追加します。



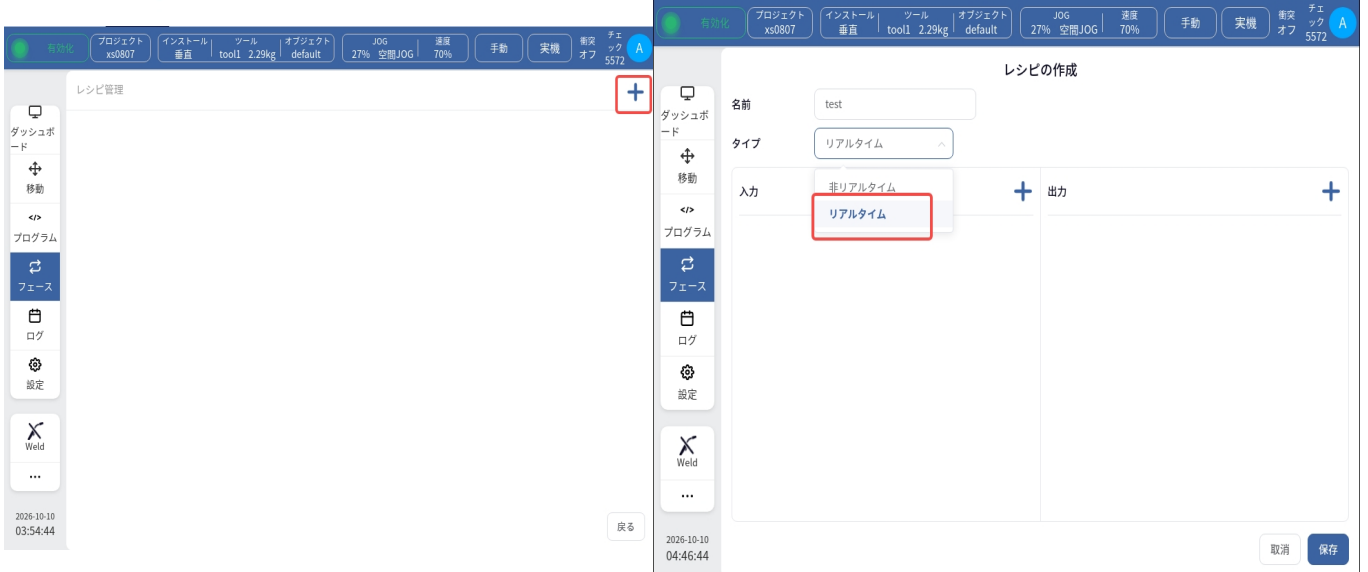
4. 出力データ項目を追加します。



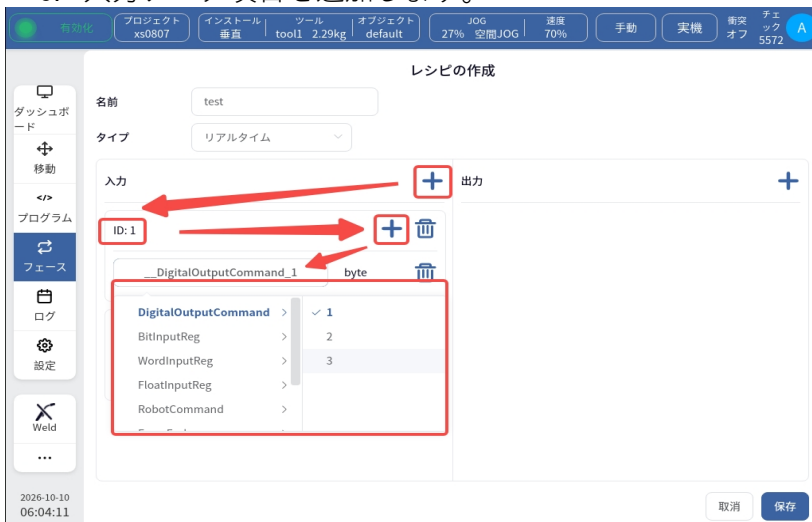
「保存」をクリックすると、レシピの作成が完了します。

システムデータレシピ

1. UI 画面で CAN ポートのレシピ管理を選択します。
2. 「+」をクリックし、レシピ名を入力して、リアルタイムレシピを選択します

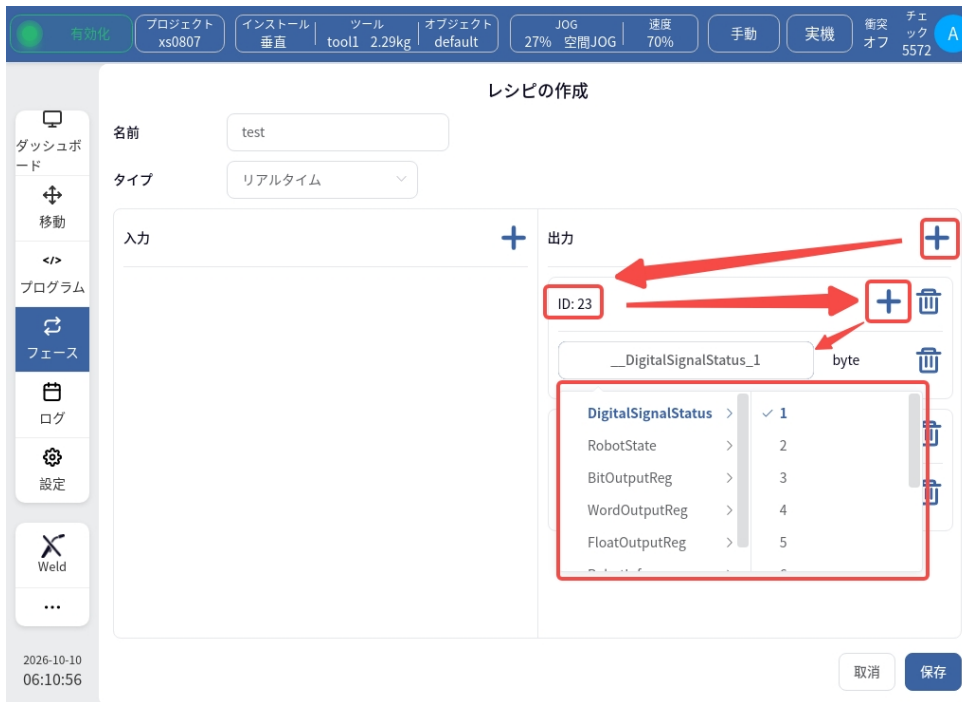


3. 入力データ項目を追加します。



3.2. レシピの作成

4. 出力データ項目を追加します。



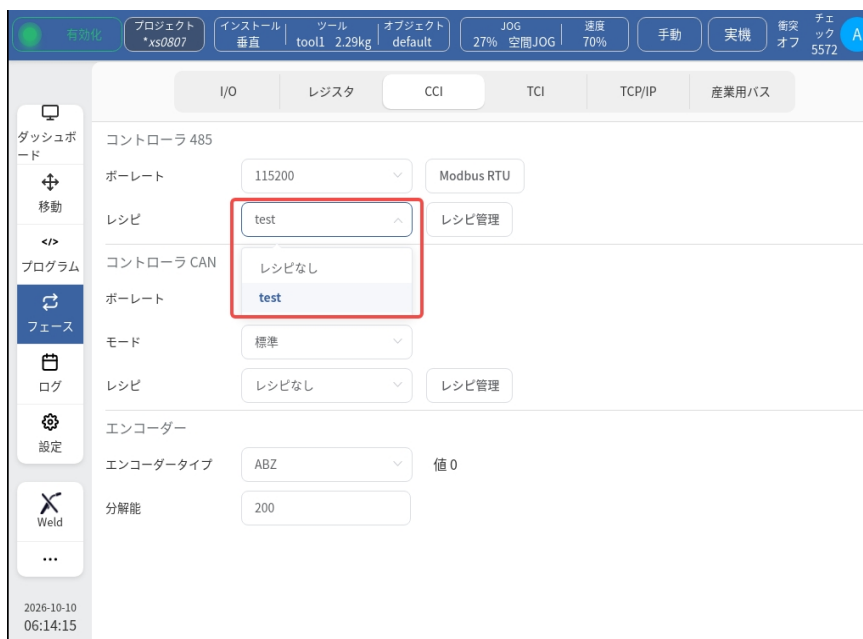
5. 「保存」をクリックすると、レシピの作成が完了します。

3.3 レシピの使用

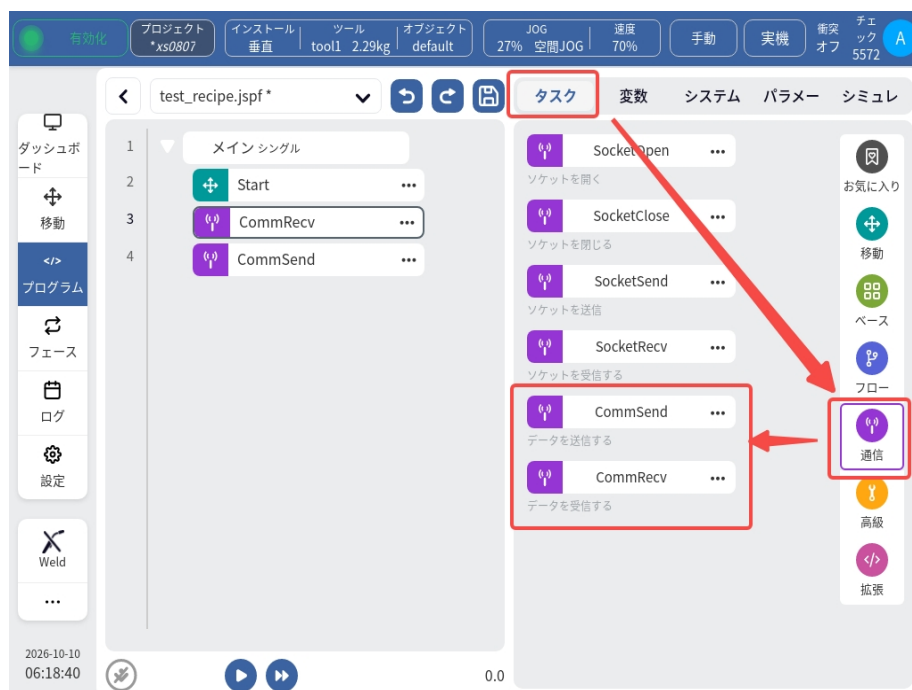
3.3.1 485 ポートレシピの使用

カスタムレシピの使用

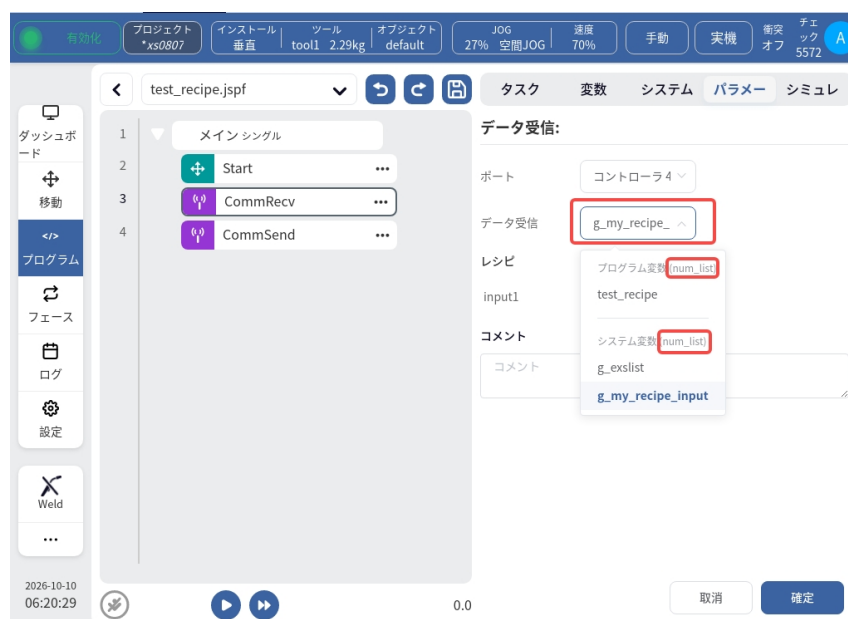
1. RS-485 変換器とシリアル通信テストツールを使用して、PC をロボットの 485 ポートに接続します。
2. 485 ポートのレシピファイルを選択します。



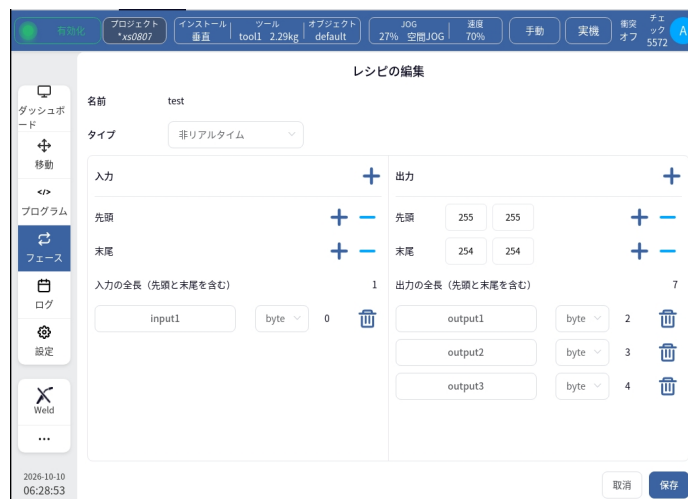
3. 処理を制御するプログラムを作成します



CommRecv:



CommSend:

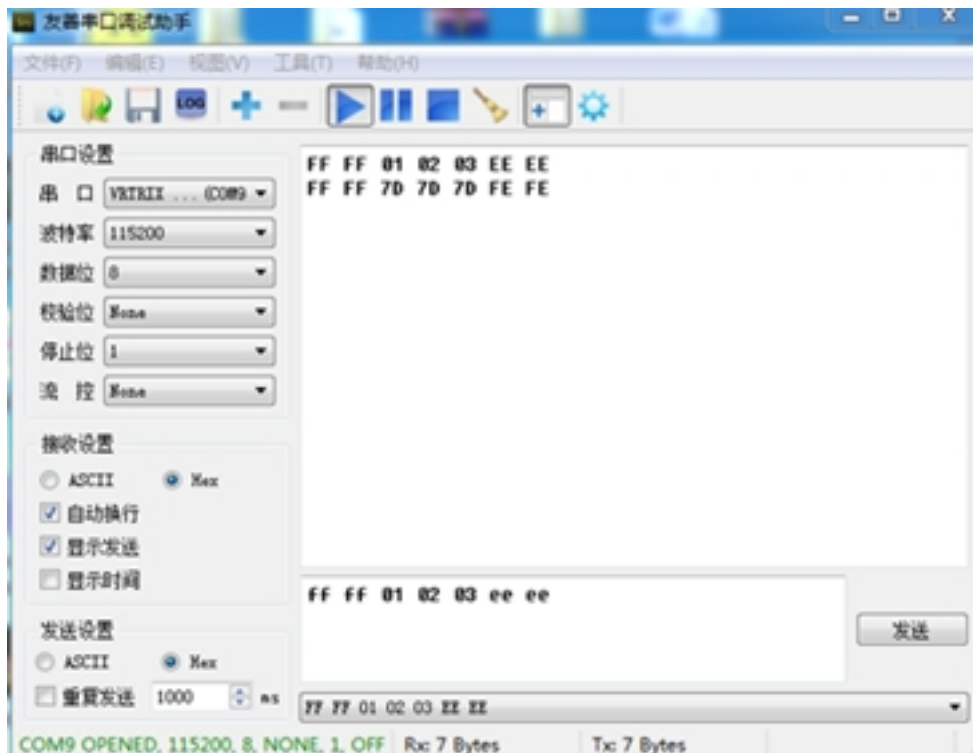




4. プログラムを実行し、シリアル通信テストツールから FF FF 01 02 03 EE EE を送信します。
5. プログラム変数画面で値を確認します。

程序変量		
名称	类型	当前值
list1	num_list	{1,2,3}

6. シリアル通信テストツールで FF FF 7D 7D 7D FE FE を受信します。

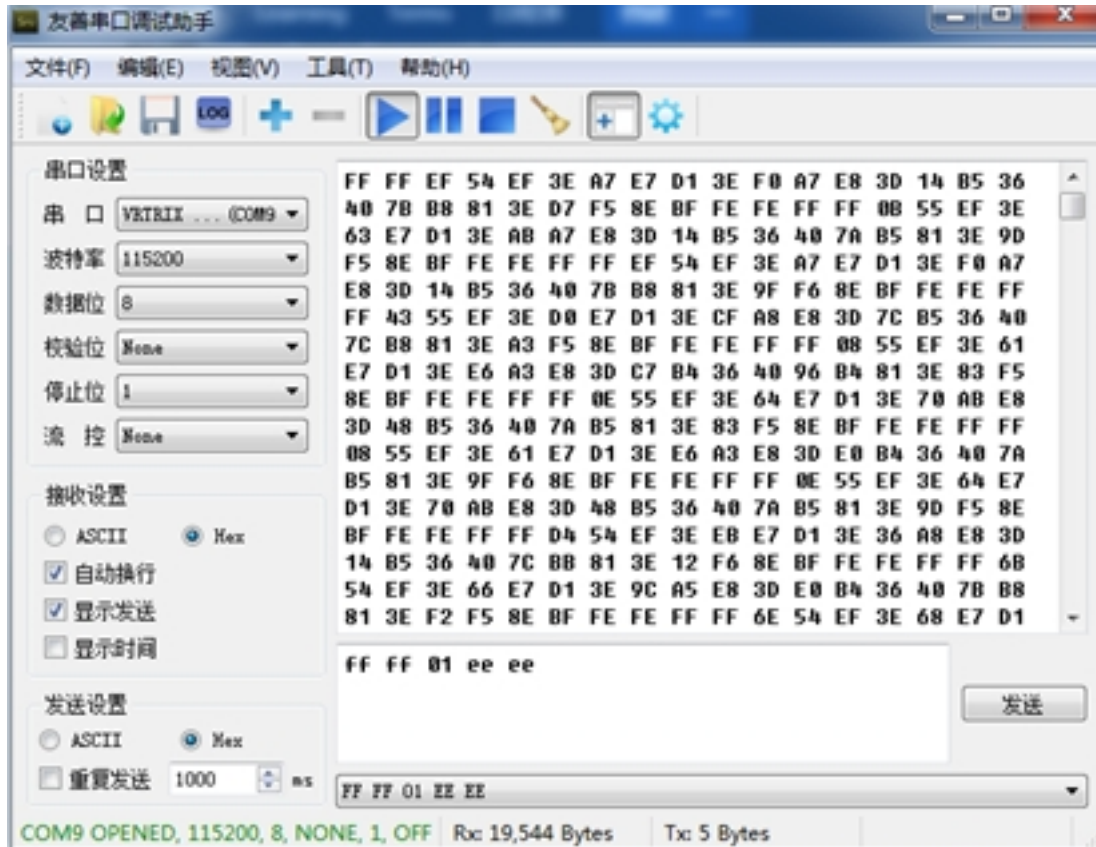


システムレシピの使用

1. RS-485 変換器とシリアル通信テストツールを使用して、PC をロボットの 485 ポートに接続します。
2. 485 ポートのレシピファイルを選択します。



3. リアルタイムシステムレシピを選択しているため、シリアル通信テストツールはロボットの位置姿勢データを周期的に受信します。



4. シリアル通信テストツールから FF FF 01 EE EE を送信すると、レシピマネージャによって対応する I/O ポートが制御されます。



3.3.2 CAN ポートレシピの使用

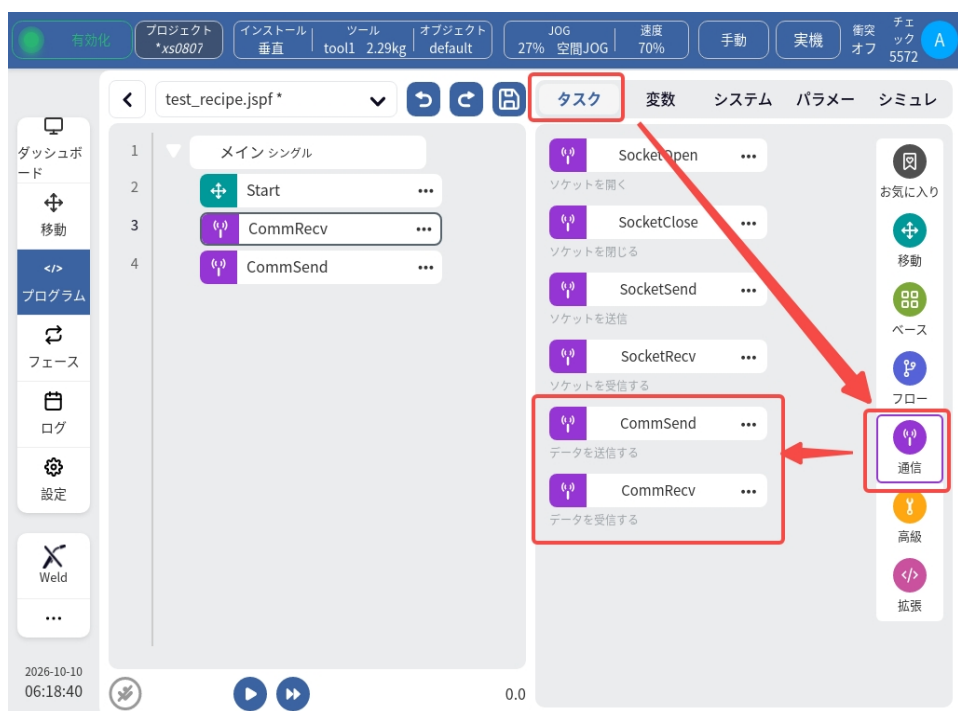
カスタムレシピの使用

1. CAN インターフェースボックスで PC とロボットを接続し、CanTest を使用して CAN 通信を確認します。
2. CAN ポートのレシピファイルを選択します。

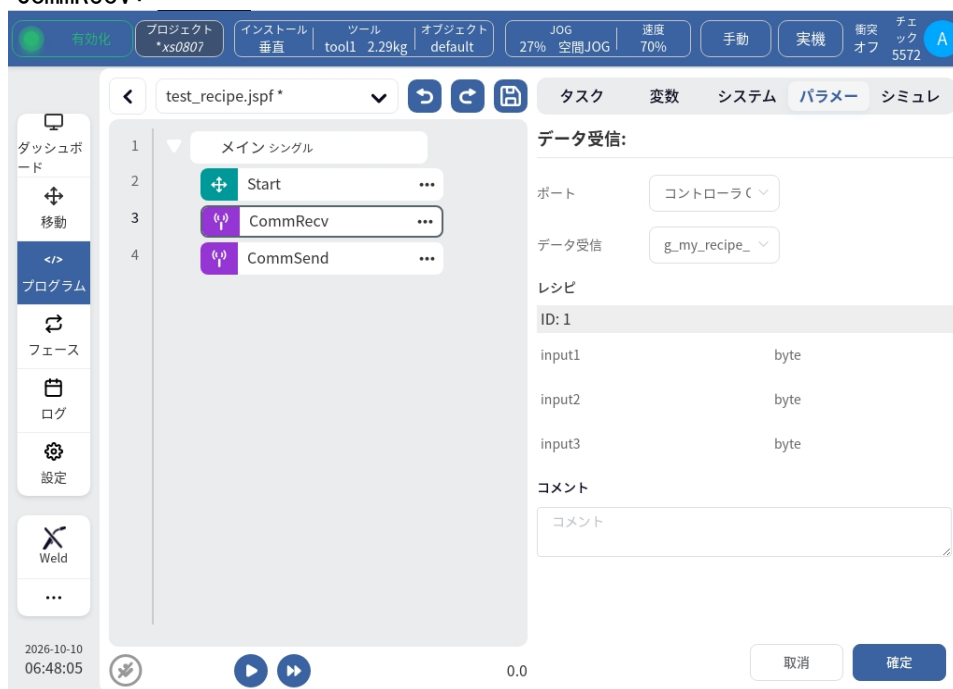


3. 処理を制御するプログラムを作成します

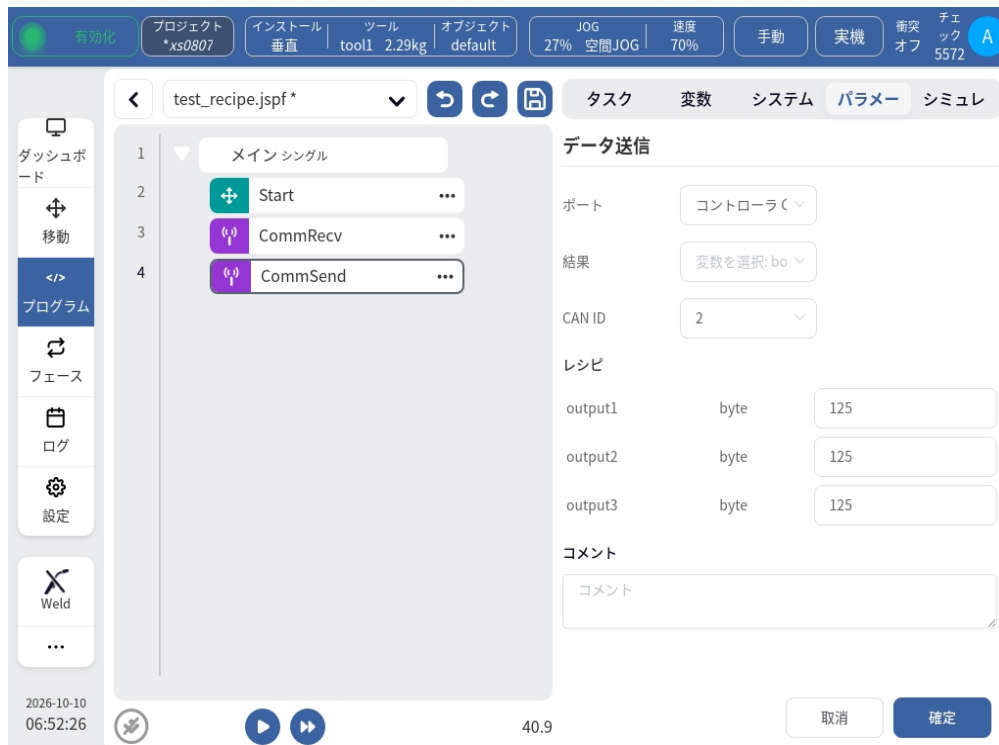
3.3. レシピの使用



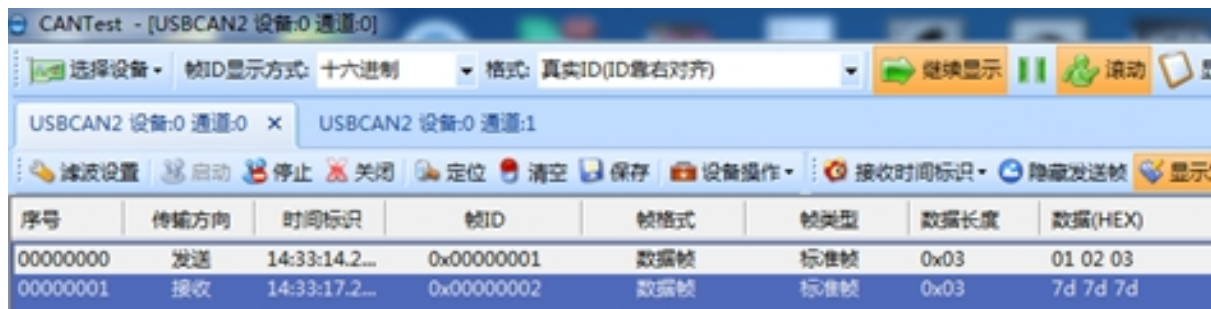
CommRecv:



CommSend:

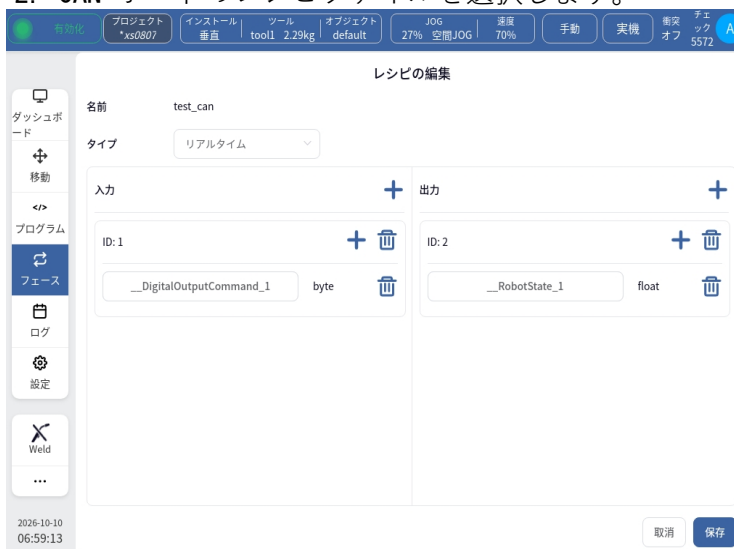


4. プログラムを実行し、CanTest からフレーム ID 2、データ 7D 7D 7Dを送信します
リストの先頭の変数は受信したデータフレームの ID で、それ以降がデータです。
5. CanTest でフレーム ID 2、データ 7D 7D 7D (125 125 125) を受信します。



システムレシピの使用

1. CAN インターフェイスボックスで PC とロボットを接続し、CanTest を使用して CAN 通信を確認します。
2. CAN ポートのレシピファイルを選択します。



3.3. レシピの使用



リアルタイムシステムレシピを選択しているため、CanTest はロボットの位置姿勢データを周期的に受信します。

序号	传输方向	时间标识	帧ID	帧格式	帧类型	数据长度	数据(HEX)
00000000	接收	14:29:48.4...	0x00000002	数据帧	标准帧	0x08	ef 54 ef 3e a7 e7 d1 3e
00000001	接收	14:29:48.4...	0x00000003	数据帧	标准帧	0x08	ef 54 ef 3e a7 e7 d1 3e
00000002	接收	14:29:48.4...	0x00000004	数据帧	标准帧	0x08	ef 54 ef 3e a7 e7 d1 3e
00000003	接收	14:29:48.4...	0x00000002	数据帧	标准帧	0x08	0b 55 ef 3e 63 e7 d1 3e
00000004	接收	14:29:48.4...	0x00000003	数据帧	标准帧	0x08	0b 55 ef 3e 63 e7 d1 3e
00000005	接收	14:29:48.4...	0x00000004	数据帧	标准帧	0x08	0b 55 ef 3e 63 e7 d1 3e
00000006	接收	14:29:48.4...	0x00000002	数据帧	标准帧	0x08	ef 54 ef 3e a7 e7 d1 3e
00000007	接收	14:29:48.4...	0x00000003	数据帧	标准帧	0x08	ef 54 ef 3e a7 e7 d1 3e
00000008	接收	14:29:48.4...	0x00000004	数据帧	标准帧	0x08	ef 54 ef 3e a7 e7 d1 3e
00000009	接收	14:29:48.4...	0x00000003	数据帧	标准帧	0x08	0e 55 ef 3e 64 e7 d1 3e
00000010	接收	14:29:48.4...	0x00000004	数据帧	标准帧	0x08	0e 55 ef 3e 64 e7 d1 3e
00000011	接收	14:29:48.4...	0x00000002	数据帧	标准帧	0x08	ec 54 ef 3e a5 e7 d1 3e
00000012	接收	14:29:48.4...	0x00000004	数据帧	标准帧	0x08	ec 54 ef 3e a5 e7 d1 3e
00000013	接收	14:29:48.4...	0x00000002	数据帧	标准帧	0x08	ef 54 ef 3e a7 e7 d1 3e
00000014	接收	14:29:48.4...	0x00000003	数据帧	标准帧	0x08	ef 54 ef 3e a7 e7 d1 3e
00000015	接收	14:29:48.4...	0x00000004	数据帧	标准帧	0x08	ef 54 ef 3e a7 e7 d1 3e
00000016	接收	14:29:48.5...	0x00000002	数据帧	标准帧	0x08	77 55 ef 3e e9 e7 d1 3e
00000017	接收	14:29:48.5...	0x00000003	数据帧	标准帧	0x08	77 55 ef 3e e9 e7 d1 3e
00000018	接收	14:29:48.5...	0x00000004	数据帧	标准帧	0x08	77 55 ef 3e e9 e7 d1 3e
00000019	接收	14:29:48.5...	0x00000002	数据帧	标准帧	0x08	0b 55 ef 3e 63 e7 d1 3e

CanTest からフレーム ID 1、データ 01 を送信すると、レシピマネージャによって対応する I/O ポートが制御されます。

发送方式: 正常发送

☒ 每次发送单帧 ☐ 每次发送 10 帧 ☐ 帧ID每发送一帧递增

帧类型: 标准帧 帧ID (HEX): 00000001 数据 (HEX): 01 发送

帧格式: 数据帧 发送次数: 1 每次发送间隔 (ms): 0 停止



3.3.3 TCP 及 UDP ポートレシピの使用

TCP/IP インターフェースでもストリームレシピインターフェースを使用できます。ただし、従来仕様との互換性により、TCP と UDP で使用できるのはシステムデータレシピ（リアルタイムレシピ）のみです。

ロボットコントローラは、TCP/IP インターフェースでレシピを使用するための次の 2 つの方式を提供します。

1. UDP server (ポート番号：2011)
2. TCP/UDP client (IP アドレスとポートはユーザーが設定)

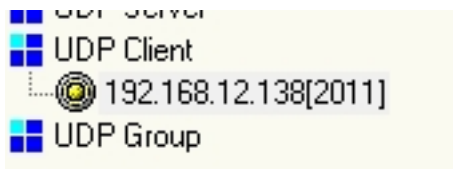
Udp server

1. LAN ケーブルで PC とコントローラを接続し、ネットワークデバッグツールでテストします
2. UDP server にレシピファイルを設定します。



3. 通信テストツールで UDP client を作成します。

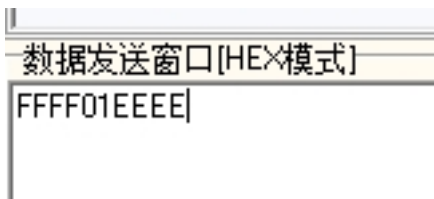
3.3. レシピの使用



- 接続に成功すると、ロボットの位置姿勢データを周期的に受信します。

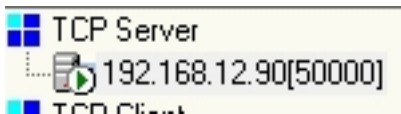


- server へ FF FF 01 EE EE を送信すると、設定した I/O が制御されます。

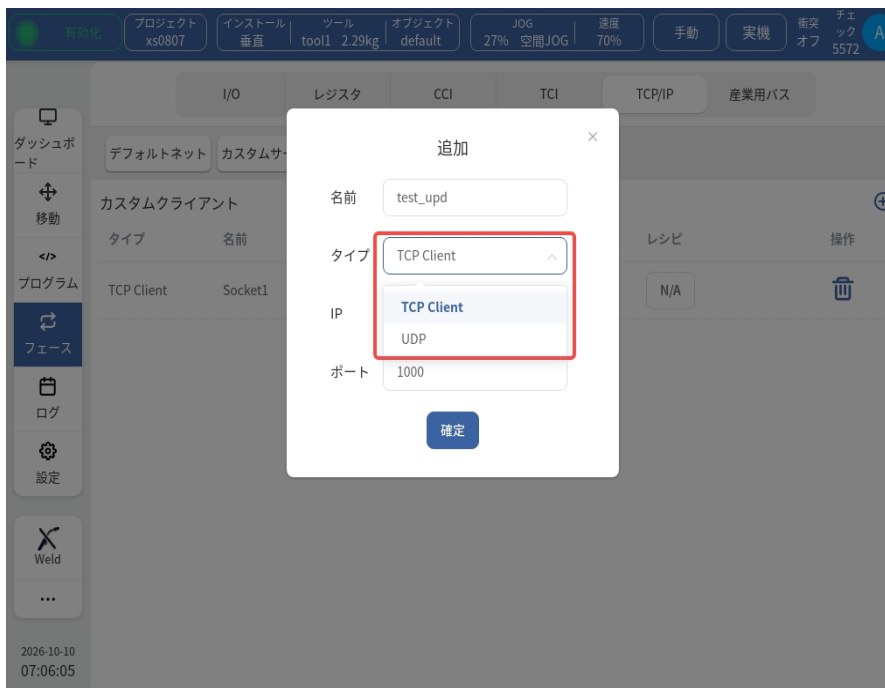


tcp/udp client

- LAN ケーブルで PC とコントローラを接続し、ネットワークデバッグツールでテストします
- 通信テストツールで TCP/UDP server を作成します。



- UI から TCP/UDP client を作成します。



4. client のレシピファイルを選択します。



5. 通信テストツールでロボットの位置姿勢データを周期的に受信します。



6. client へ FF FF 01 EE EE を送信すると、設定した I/O が制御されます。

PROFINET DEVICE インターフェース操作マニュアル

4.1 Profinet device インターフェース設定

4.1.1 インターフェースの説明

ハードウェアインターフェースについては、『DUCO CORE ハードウェアマニュアル』を参照してください。

PROFINET データ交換の推奨周期は 16 ms、Watchdog 設定値は 5 です。最短通信周期は 8 ms 未満に設定しないでください。

4.1.2 スクリプト関数

ロボットには、データインターフェースの値を読み書きするためのスクリプト関数が用意されています。

```
write_reg(number:num, number:type, val)
```

関数の説明：

この関数は内部レジスタの値を変更します。

パラメータ：

num：内部レジスタ番号。

type：変更するレジスタの種類。1 は bool レジスタ、2 は word レジスタ、3 は float レジスタです。

val：val の型は type に従って決まります。

type が 1 の場合、val の型は boolean です。true は真、false は偽を表し、num の範囲は 1～64 です。

type が 2 の場合、val の型は number で、値の範囲は 0～65535、num の範囲は 1～32 です。

type が 3 の場合、val の型は number で、num の範囲は 1～32 です。

パラメータが不正な場合、内部レジスタの値は変更されません。

戻り値：

なし

例：

```
write_reg( 5, 1, true)
```

```
read_reg (number:num, number:type, number:in_out)
```

関数の説明：

この関数は内部レジスタの値を読み出します。

パラメータ：

num：内部 bool レジスタ番号。

type：レジスタの種類。1 は bool レジスタ、2 は word レジスタ、3 は float レジスタです。

type が 1 の場合、num の範囲は 1～64 です。

type が 2 の場合、num の範囲は 1～32 です。

type が 3 の場合、num の範囲は 1～32 です。

パラメータが不正な場合、内部レジスタの値は変更されません。

in_out：0 は入力レジスタの読み出し、1 は出力レジスタの読み出しを表します。

戻り値：

type が 1 の場合、戻り値の型は boolean です。true は真、false は偽を表します。

type が 2 の場合、戻り値の型は number で、範囲は 0～65535 です。

type が 3 の場合、戻り値の型は number です。

例：

```
ret=read_reg(10, 1, 1)
```

4.1.3 設定手順

以下では Siemens PLC を例に、TIA Portal V14 を使用した PROFINET インターフェースの設定手順を説明します。

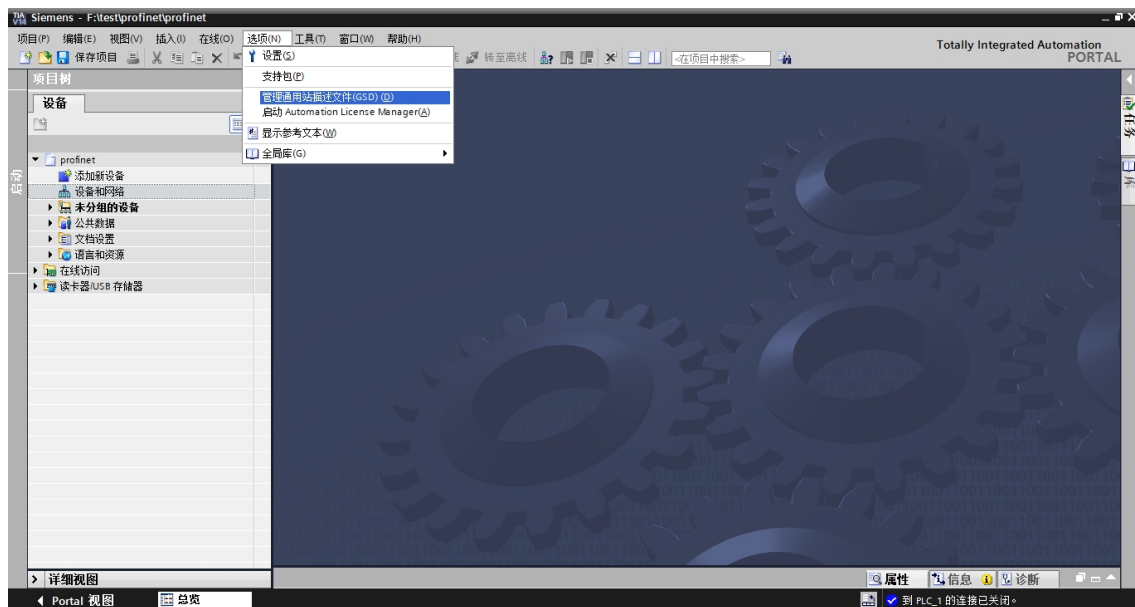
- ・新規プロジェクトを作成する場合は“☒建新☒目”をクリックします。既存プロジェクトを開く場合は“☐打开☒有☒目”をクリックします。



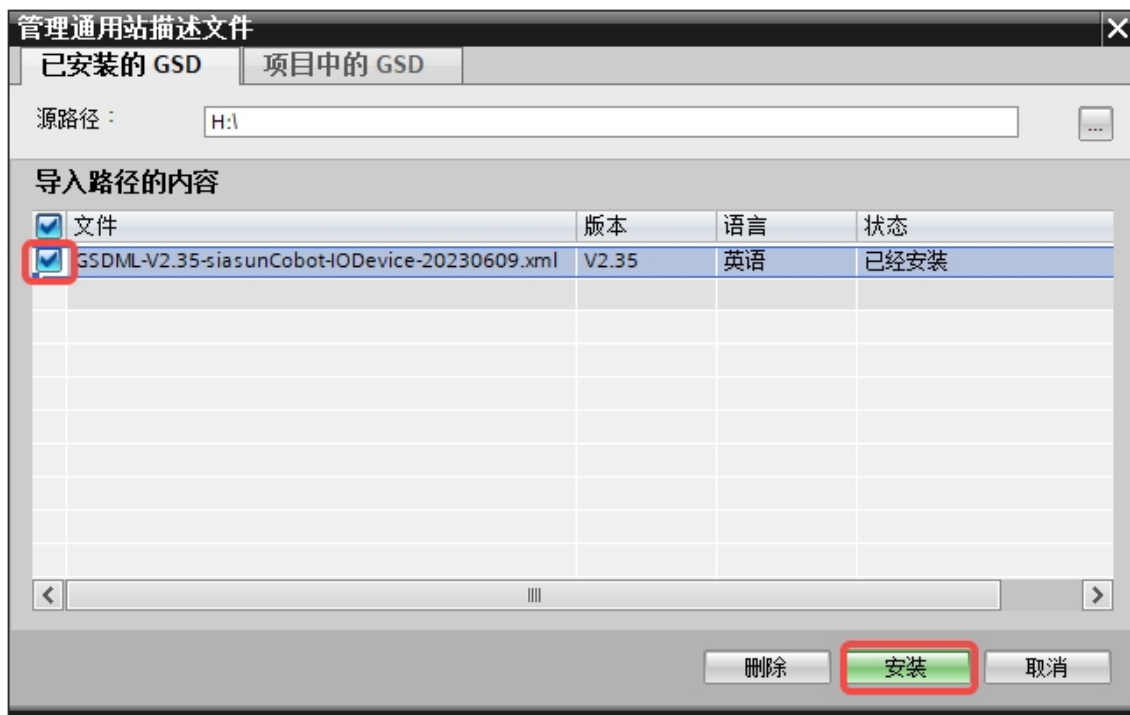
- ・ “打开☐目☐☐”をクリックします。



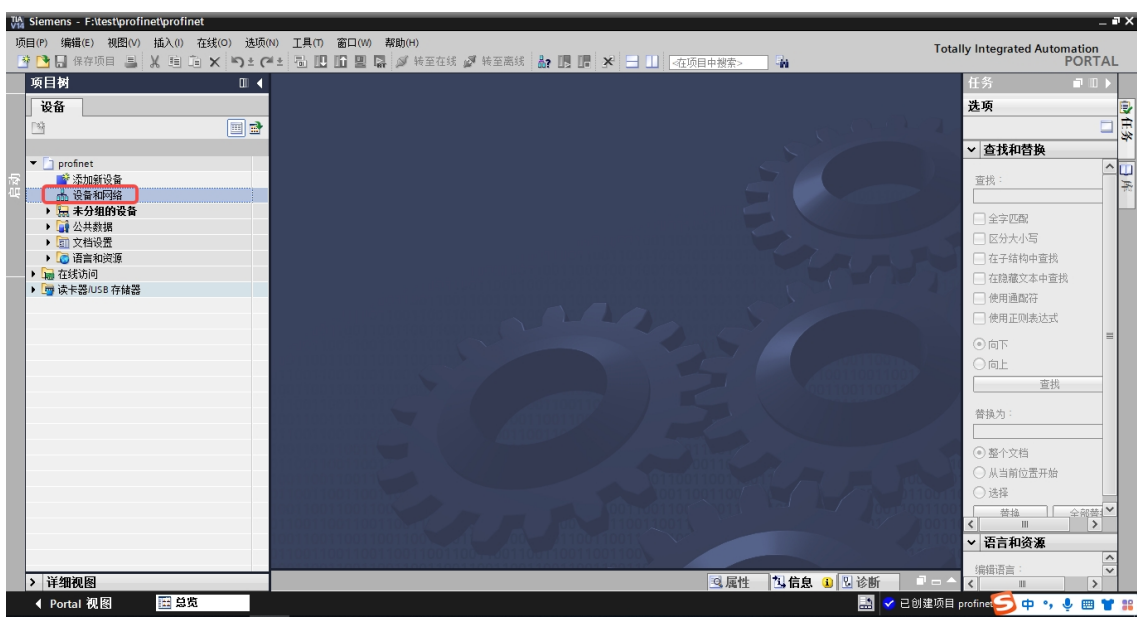
- ・ メニューバーの“☐☐”から“安装☐☐描述文件 (GSD)”をクリックします。



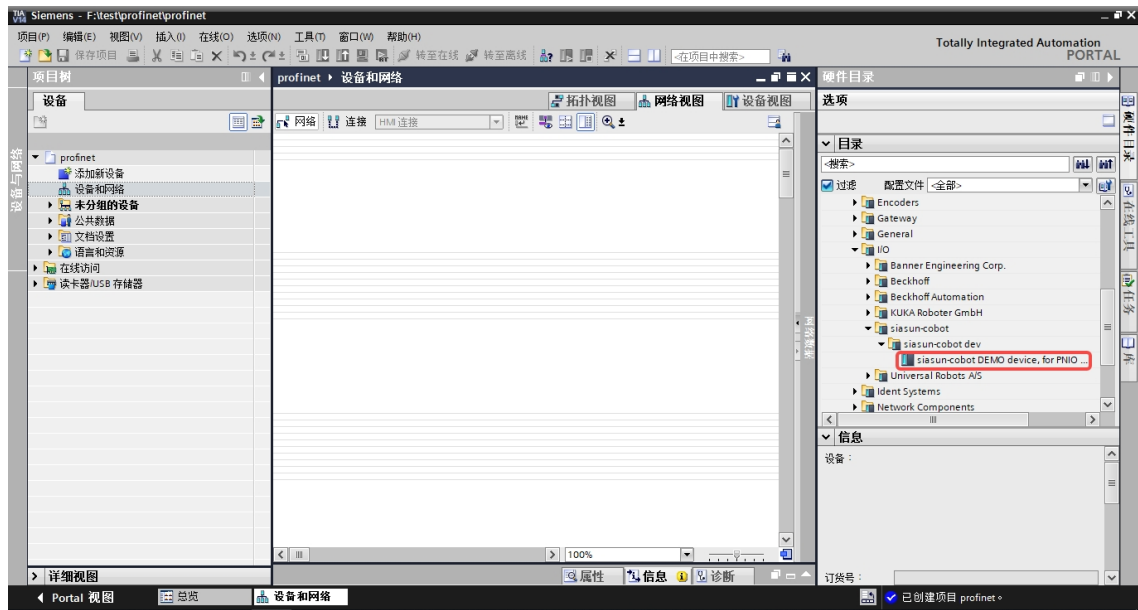
- ・ 次の図の番号順に操作します。1 のボタンをクリックし、GSD ファイルの経路（パス）を選択します。
- 2 のチェックボックスで、インポートする GSDML-V2.35-siasunCobot-IIODevice-20200615.xml ファイルを選択し、最後に 3 の“安装”をクリックします。



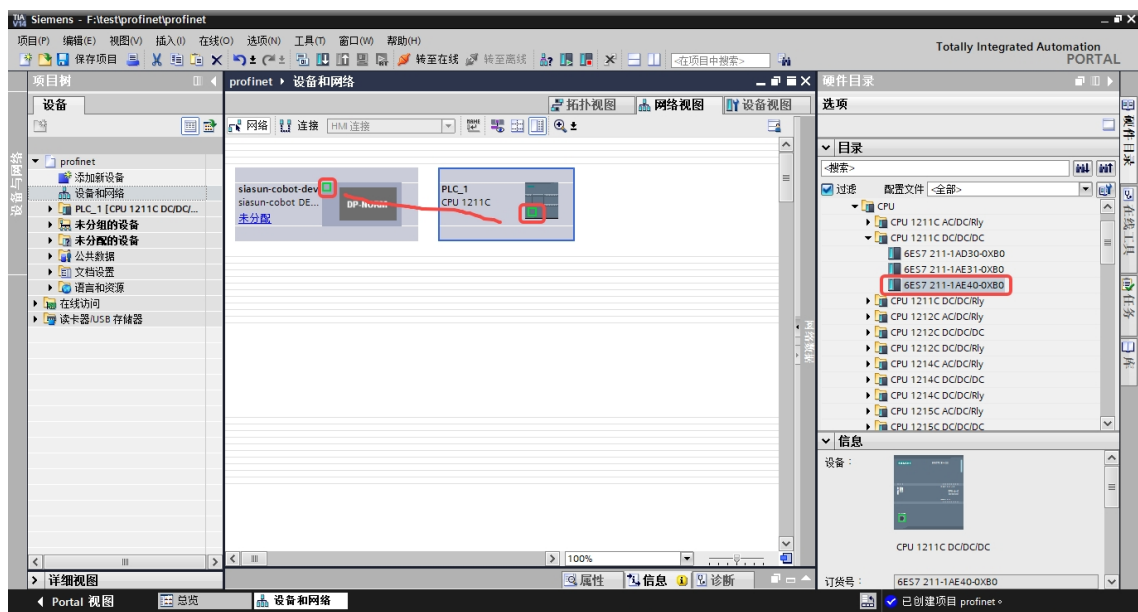
- “☒ ☒ 和 网 ☒”をダブルクリックし、左端の列で“硬件目 ☒”を選択します。



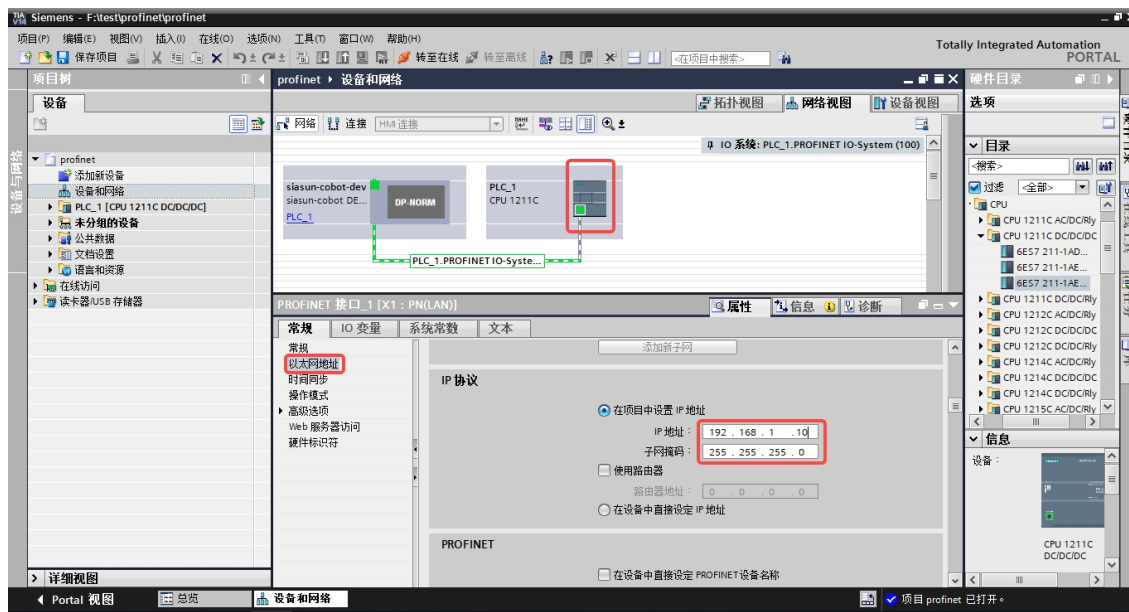
- “其他 ☒ ☒ ☒ ☒ -> PROFINET IO -> I/O -> siasun-cobot dev”の下にある
“siasun-cobot Demo dev”をダブルクリックします。バージョンには、直前に読み込んだ
GSDML-V2.35-siasunCobot-IODevice-20200615.xml



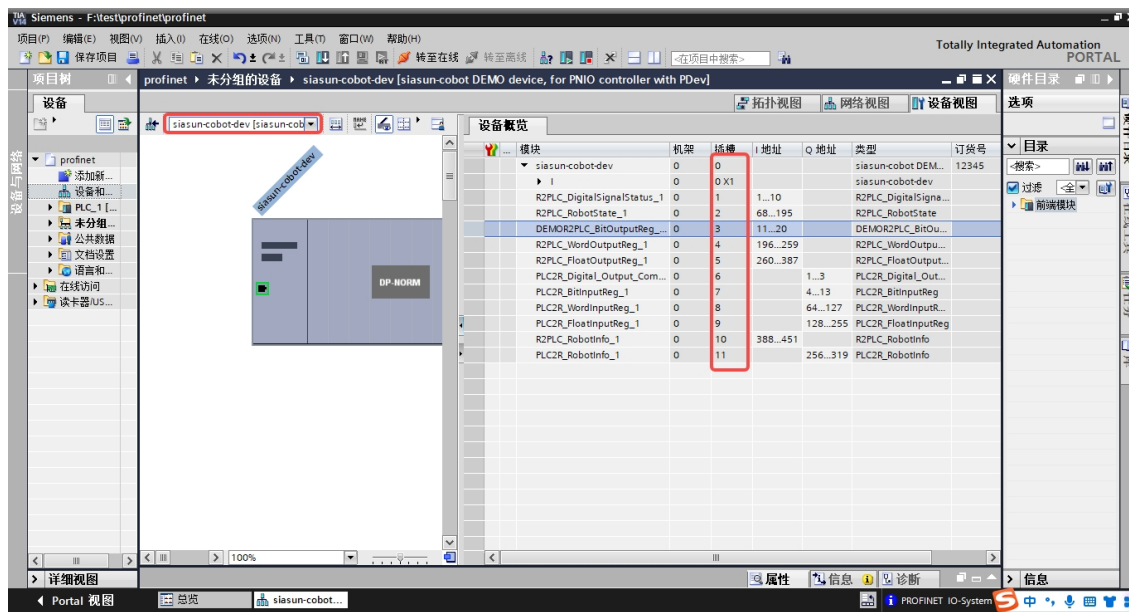
- PROFINET Device の構成にはコントローラも必要です。前述と同じ方法で CPU1211C DC/DC/DC をコントローラとして追加します。一方のデバイスの緑色の四角を左クリックしたまま他方のデバイスの緑色の四角までドラッグし、デバイスを接続します。



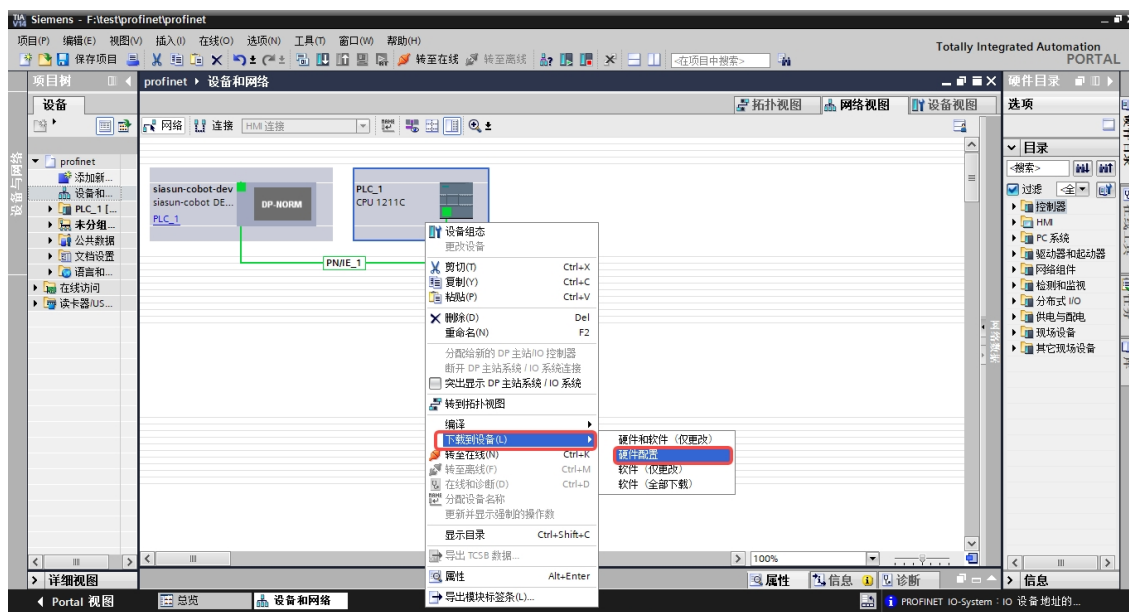
- デバイスを選択し、“属性->PROFINET 接口-> 以太网地址”で“在☑目中☑置 IP 地址”を選択して、“IP 地址”と“子网掩☑”を入力します。もう一方のデバイスも同様に設定します。



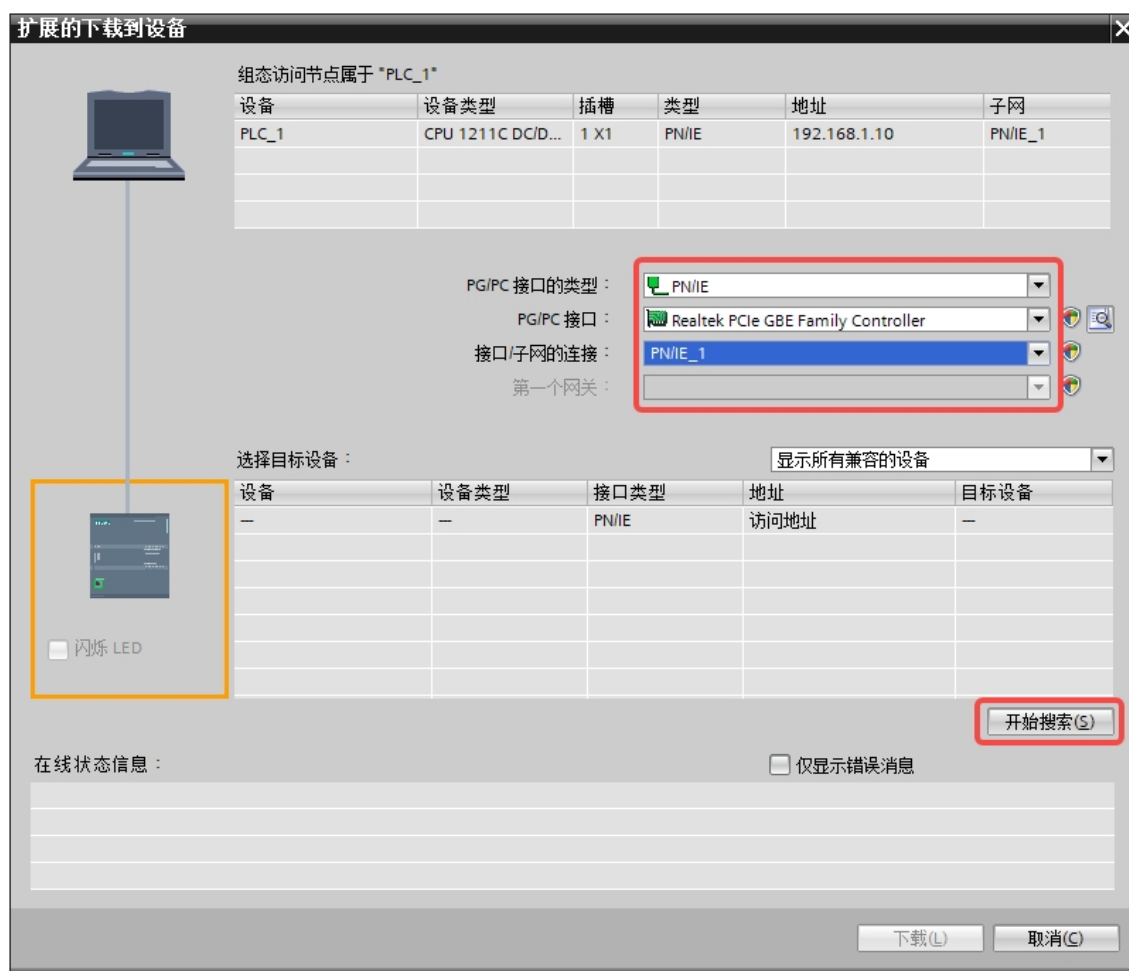
- ・“siasun-cobot dev”デバイスの“属性->PROFINET 接口-> 高[]-[]-> []-[]-[]-[]-> I/O 周期”で“手[]-[]-[]-[]”を選択し、更新時間を 8 ms に設定します。
- ・“[]-[]-[]-[]”をクリックします。
- ・“siasun-cobot dev”デバイスを選択し、番号の異なるスロット内で“模[]”の各変数型をダブルクリックしてモジュールを追加します。



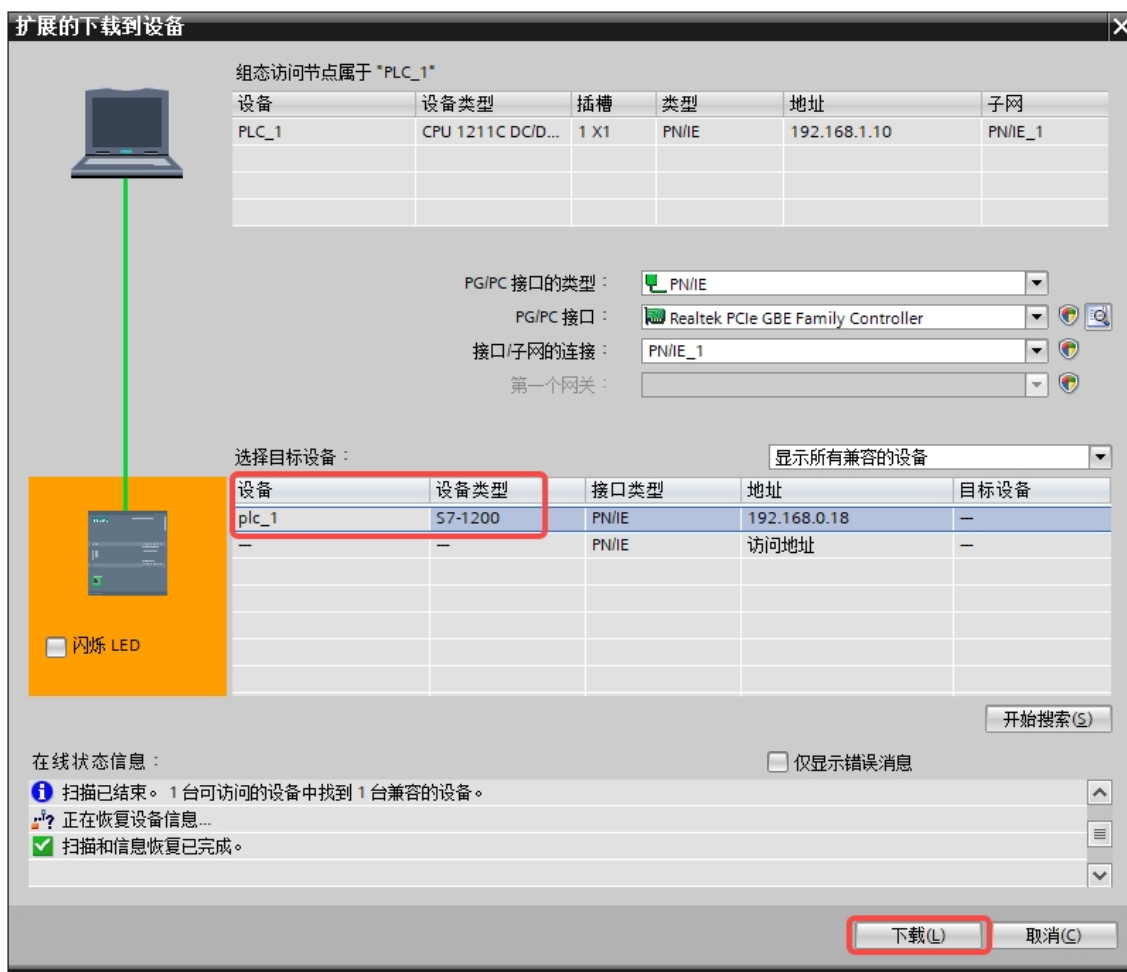
- ・各スロットのデータ型は次の図のとおりです。詳細は後述します。
- ・追加が完了したら、コントローラ（ここでは CPU1211C）を右クリックし、“下[]-[]-[]-[]”の“硬件配置”を選択します。



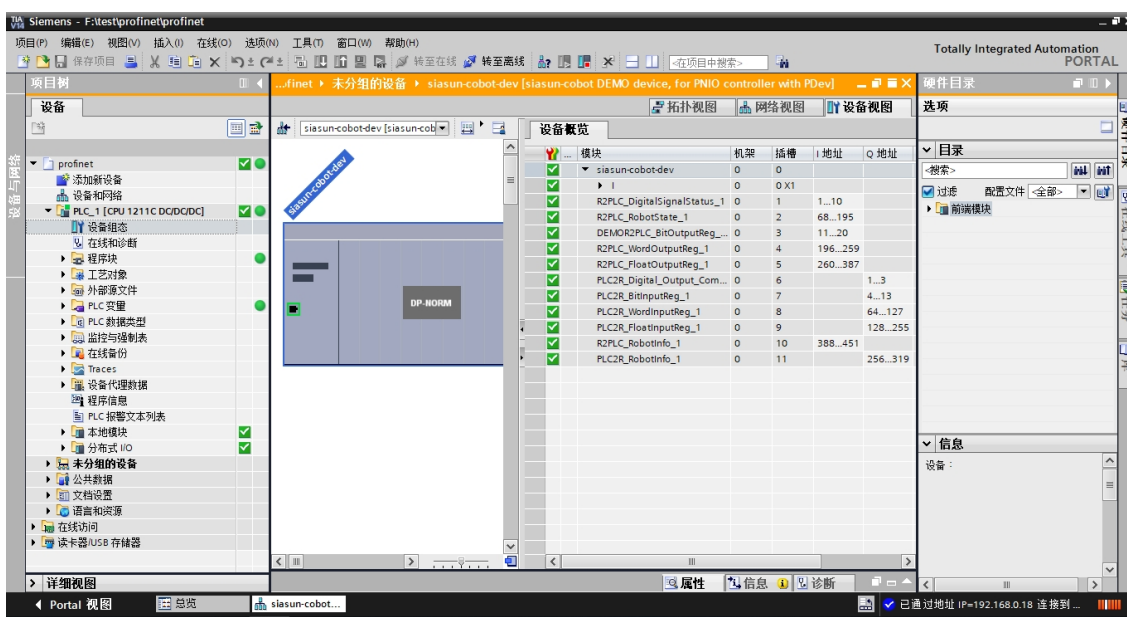
- ・ “PG/PC 接口的类型”、“PG/PC 接口”、および“インターフェース/子網の☒接”を選択し、“开始搜索”をクリックします（TIA を実行する設定用 PC からコントローラへ接続できる必要があります。たとえば、同一サブネットに接続してください）。



- ・ 検索されたコントローラ（ここでは CPU 1121C）を選択し、“下☒”をクリックします。



- ・ 問題がなければ“下”をクリックし、ダウンロードの完了を待ちます。
- ・ “完成”をクリックし、起動を待ちます。
- ・ “到在”をクリックします。“断”タブにエラーが表示されていないければ、通信は正常です。



- ・ 通信が正常な場合、すべてのモジュールは“色底白色勾”で表示されます。

4.2 Slot データ構造

4.2.1 Slot データ構造

注：各スロット (slot) の WORD 型および FLOAT 型データは、バイトオーダー変換済みです (下位バイトと上位バイトを入れ替えています)。

(1) Slot1_R2PLC_DigitalSignalStatus (10 byte)：ロボットデジタル信号状態 データ方向：ロボット——》PLC

	bit0	bit1	bit2	bit3	bit4	bit5	bit6	bit7
BYTE1	general_digital_input[0..7]							
BYTE2	general_digital_input[8..15]							
BYTE3	tool_digital_input[0..7]							
BYTE4	general_digital_output[0..7]							
BYTE5	general_digital_output[8..15]							
BYTE6	tool_digital_output[0..7]							
BYTE7	fun_digital_input[0..7]							
BYTE8	fun_digital_output[0..7]							
BYTE9	safety_state_input							
BYTE10	safety_state_output							

端子の定義については、『DUCO CORE ハードウェアマニュアル』を参照してください。

general_digital_input：制御盤汎用 DI 状態。

tool_digital_input：ロボットアーム末端 DI 状態

general_digital_output：制御盤汎用 DO 状態

tool_digital_output：ロボットアーム末端 DO 状態

fun_digital_input：制御盤機能 DI 状態

fun_digital_output：制御盤機能 DO 状態

安全コントローラ状態 safety_state の構造

	safety_state_1	safety_state_2
bit0	system_emergency_stop	config_safety_output0
bit1	external_emergency_stop	config_safety_output1
bit2	protective_stop_input	reserved
bit3	operation_mode_input	reserved
bit4	3_position_enable_input	reserved
bit5	config_safety_input0	reserved
bit6	config_safety_input1	reserved
bit7	reserved	reserved

config_safety_input：画面で設定した安全入力

config_safety_output：画面で設定した安全出力

(2) Slot2_R2PLC_RobotState (29 float)：ロボット状態情報出力 データ方向：ロボット——》PLC

Float1..7	joint_pos (rad)
Float8..13	tcp_pose (ベース座標系基準の tcp 値)
Float14..19	tcp_force
Float20..25	tcp_offset
Float26..29	tcp_load (重心、質量)

(3) Slot3_R2PLC_BitOutputReg (10 byte) : ビット出力レジスタ。ロボットの現在のビットレジスタ状態を出力 データ方向：ロボット——》PLC

	bit0	bit1	bit2	bit3	bit4	bit5	bit6	bit7
BYTE1	fun_registers_output[1..8]							
BYTE2	fun_registers_output[9..16]							
BYTE3	bool_registers_output[1..8]							
BYTE4	bool_registers_output[9..16]							
BYTE5	bool_registers_output[17..24]							
BYTE6	bool_registers_output[25..32]							
BYTE7	bool_registers_output[33..40]							
BYTE8	bool_registers_output[41..48]							
BYTE9	bool_registers_output[49..56]							
BYTE10	bool_registers_output[57..64]							

(4) Slot4_R2PLC_WordOutputReg (64 byte) :Word 出力レジスタ データ方向：ロボット——》PLC

Word1..32	word_output_register [1..32]
-----------	------------------------------

(5) Slot5_R2PLC_FloatOutputReg (32 float) : 浮動小数点出力レジスタ データ方向：ロボット——》PLC

Float1..32	float_output_register [1..32]
------------	-------------------------------

(6) Slot6_PLC2R_Digital_Output_Command (3 byte) : ロボット制御コマンド入力 データ方向：PLC——》ロボット

	bit0	bit1	bit2	bit3	bit4	bit5	bit6	bit7
BYTE1	general_digital_output[1..8]							
BYTE2	general_digital_output[9..16]							
BYTE3	tool_digital_output[1..8]							

(7) Slot7_PLC2R_BitInputReg(10 byte): 汎用ビット入力レジスタ データ方向：PLC——》ロボット

	bit0	bit1	bit2	bit3	bit4	bit5	bit6	bit7
BYTE1	fun_input_register[1..8]							
BYTE2	fun_input_register[9..16]							
BYTE3..10	bit_input_register[1..64]							

(8) Slot8_PLC2R_WordInputReg (64 byte): Word 入力レジスタ データ方向: PLC——》ロボット

Word1..32	word_input_register[1..32]
-----------	----------------------------

(9) Slot9_PLC2R_FloatInputReg (32 float): 汎用浮動小数点入力レジスタ データ方向: PLC ——》ロボット

Float1..32	float_input_register[1..32]
------------	-----------------------------

(10) Slot10_R2PLC_RobotInfo (16 float): ロボット速度などの情報 データ方向: ロボット——》PLC

Float1	グローバル速度オーバーライドを読み出す
Float2	Jog 速度オーバーライドを読み出す
Float3	TCP 合成並進速度を読み出す
Float4	シミュレーション/実機モードを読み出す
Float5	エラー ID (システムの最終エラー ID) を読み出す
Float6..16	予約

(11) Slot11_PLC2R_RobotInfo (16 float): ロボット速度などの情報 データ方向: PLC ——》ロボット

Float1	グローバル速度オーバーライドを読み出す
Float2..16	予約

ETHERNET IP インターフェース操作マニュアル

5.1 Ethernet 及 IP インターフェース設定

5.1.1 インターフェースの説明

ハードウェアインターフェースについては、『DUCO CORE ハードウェアマニュアル』を参照してください。

転送タイプ：**Exclusive Owner**。スキャナからロボットへのデータ送信と、ロボットからスキャナへのデータ受信を同時に設定できます。

トリガタイプ：**Cycle**。データ転送を周期的に実行します。

モード：**Point-to-Point**。スキャナは対象デバイスから 1 対 1 でデータを受信します。

入力経路アドレス (Input Assembly)：100 (0x64)。

出力経路アドレス (Output Assembly)：150 (0x96)。

デフォルト入力サイズ (O→T size)：276 byte、範囲は [0, 1024] です。

デフォルト出力サイズ (T→O size)：476 byte、範囲は [0, 1024] です。

通信要求パケット間隔 (RPI)：デフォルトは 20 ms、範囲は [5 ms, 1000 ms] です。

5.1.2 スクリプト関数

ロボットには、データインターフェースの値を読み書きするためのスクリプト関数が用意されています。

```
write_reg(number:num, number:type, val)
```

関数の説明：

この関数は内部レジスタの値を変更します。

パラメータ：

num：内部レジスタ番号。

type：変更するレジスタの種類。1 は **bool** レジスタ、2 は **word** レジスタ、3 は **float** レジスタです。

val：**val** の型は **type** に従って決まります。

type が 1 の場合、val の型は boolean です。true は真、false は偽を表し、num の範囲は 1～64 です。

type が 2 の場合、val の型は number で、値の範囲は 0～65535、num の範囲は 1～32 です。

type が 3 の場合、val の型は number で、num の範囲は 1～32 です。

パラメータが不正な場合、内部レジスタの値は変更されません。

戻り値：

なし

例：

```
write_reg( 5, 1, true)
```

```
read_reg (number:num, number:type, number:in_out)
```

関数の説明：

この関数は内部レジスタの値を読み出します。

パラメータ：

num：内部 bool レジスタ番号。

type：レジスタの種類。1 は bool レジスタ、2 は word レジスタ、3 は float レジスタです。

type が 1 の場合、num の範囲は 1～64 です。

type が 2 の場合、num の範囲は 1～32 です。

type が 3 の場合、num の範囲は 1～32 です。

パラメータが不正な場合、内部レジスタの値は変更されません。

in_out：0 は入力レジスタの読み出し、1 は出力レジスタの読み出しを表します。

戻り値：

type が 1 の場合、戻り値の型は boolean です。true は真、false は偽を表します。

type が 2 の場合、戻り値の型は number で、範囲は 0～65535 です。

type が 3 の場合、戻り値の型は number です。

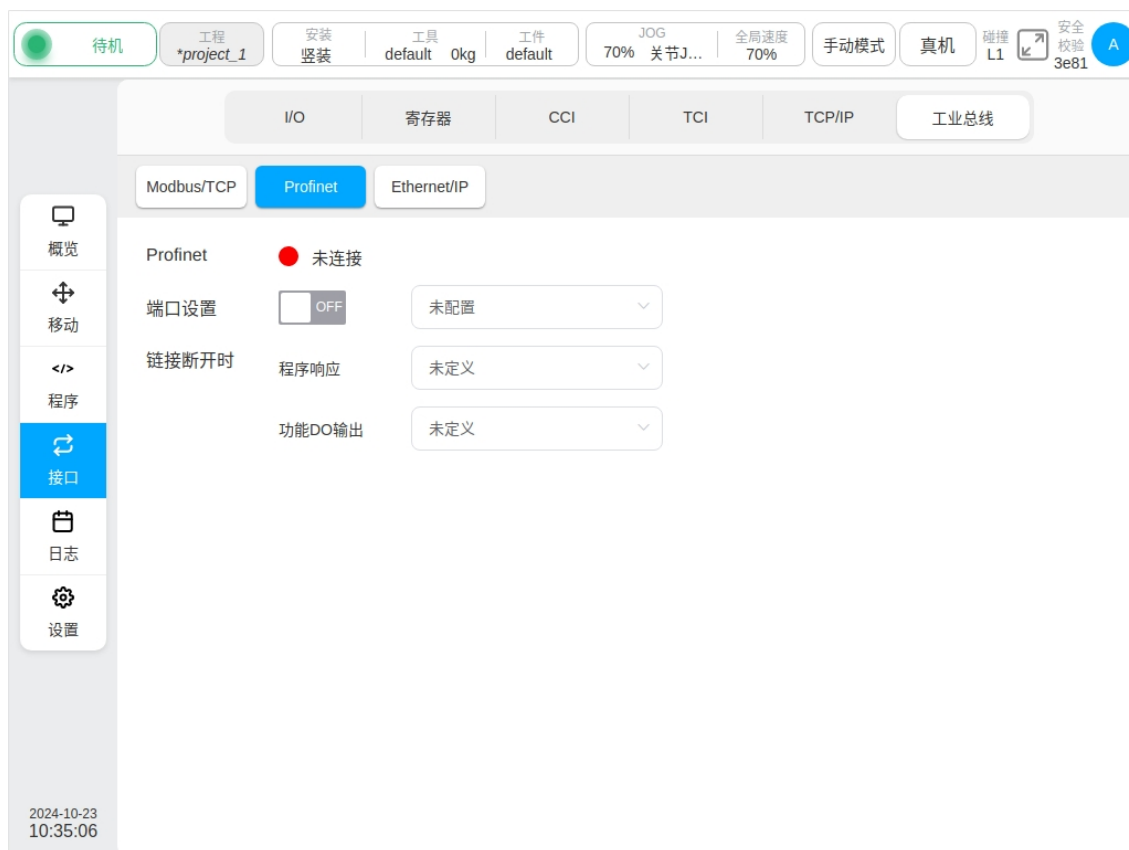
例：

```
ret=read_reg(10, 1, 1)
```

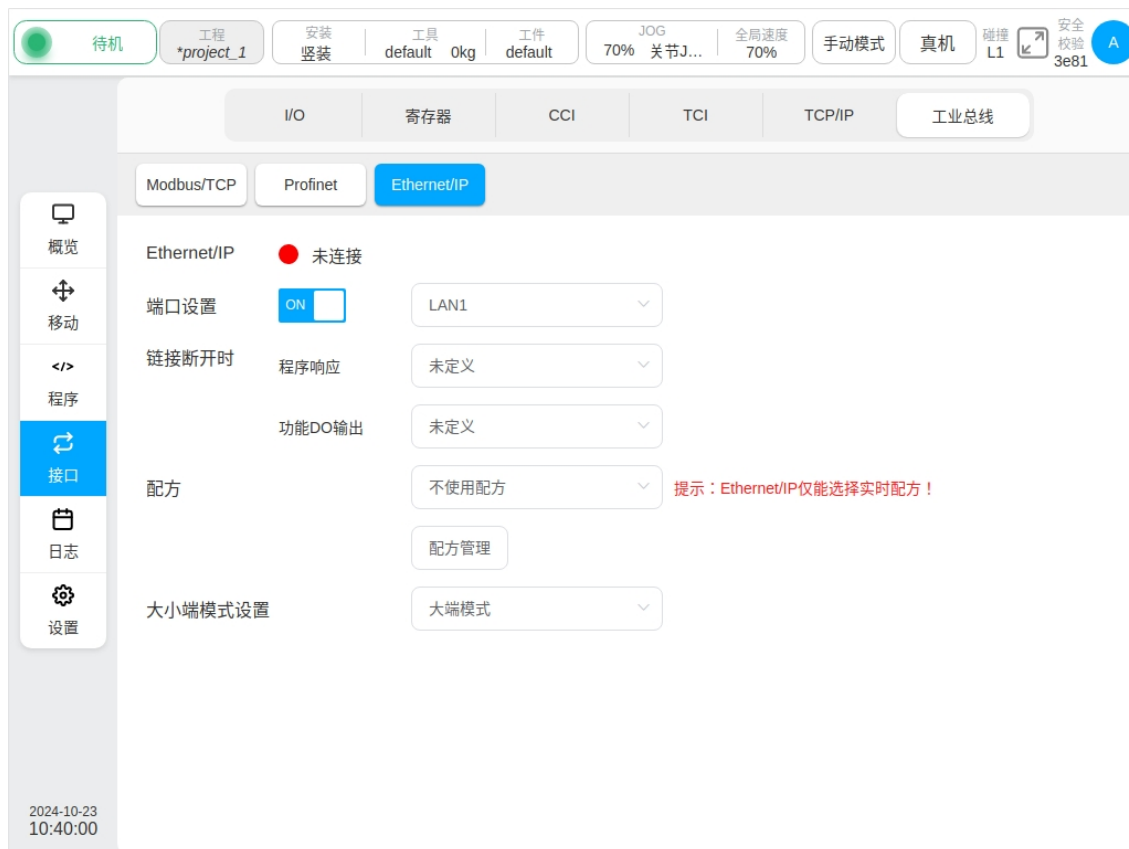
5.1.3 設定手順

ロボット（スレーブ）の設定

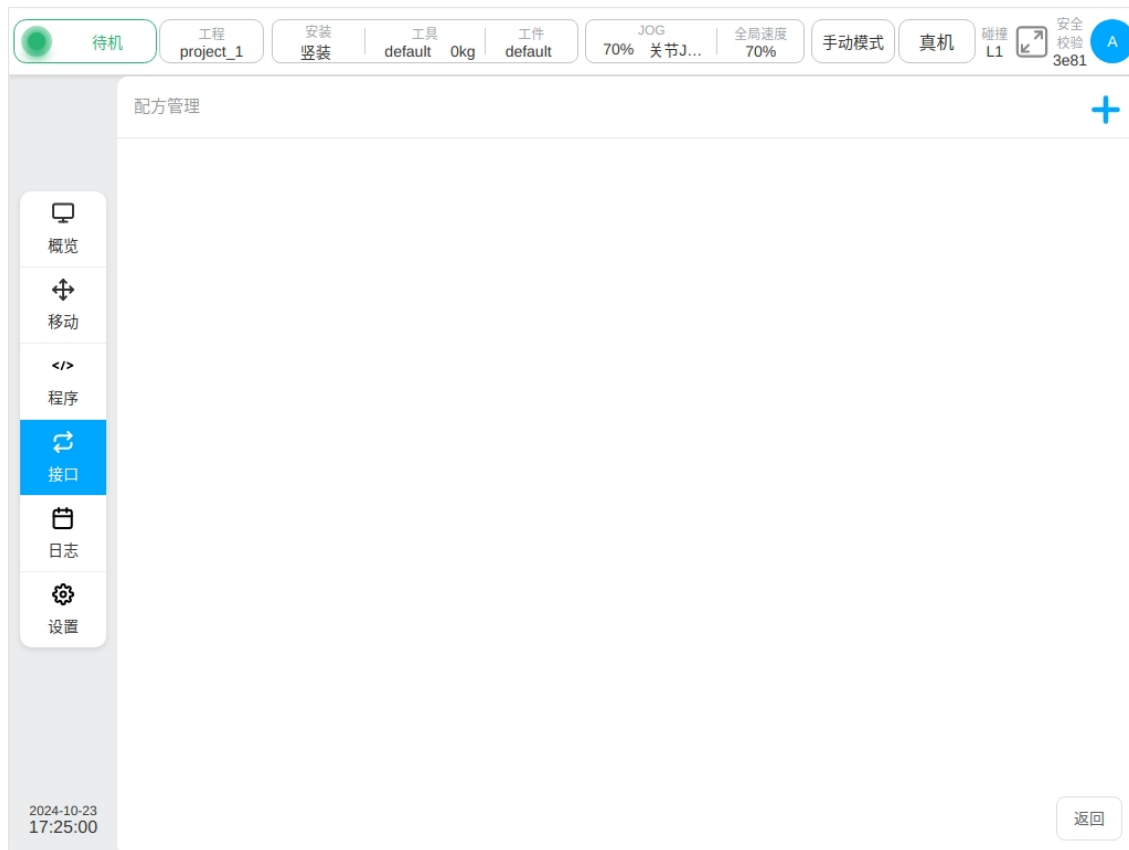
- ・ロボットコントローラの LAN1 ポートには、デフォルトで PROFINET 通信が割り当てられています。ロボット画面で“ 接口”->“ 工機”->“Profinet”の順にクリックし、“ 端口置”で“ 不配置”を選択して、有効化スイッチを“OFF”にします。



- ・ “EtherNet/IP” をクリックし、“ 端口☑置” で“LAN1” を選択して、有効化スイッチを“ON” にします。



- ・ ロボットの EtherNet/IP 通信では、デフォルトでレシピを使用しません。入出力データ構造については 5.2 章を参照してください。ロボット画面で任意のデータサイズおよびデータ構造を設定することもできます。
- ・ “ 配方管理 ”、続いて “+” アイコンをクリックします。表示されたレシピ設定画面でレシピ名を入力し、“ 配方 ” を選択します。EtherNet/IP で使用できるのはリアルタイムレシピのみです。



新建配方

配方名称: test1

配方类型: 实时配方

输入数据	输出数据
帧头	帧头
帧尾	帧尾
输入数据总长度(含帧头帧尾)	输出数据总长度(含帧头帧尾)
0	0

取消 保存

2024-10-23 17:30:00

- ・ “+” アイコンをクリックし、入力および出力データ項目を追加します。データ項目の意味については第 3 章を参照してください。たとえば、9 byte の入力データと 20 byte の出力データを追加し、“保存”、“返回”の順にクリックします。

编辑配方

配方名称: test1

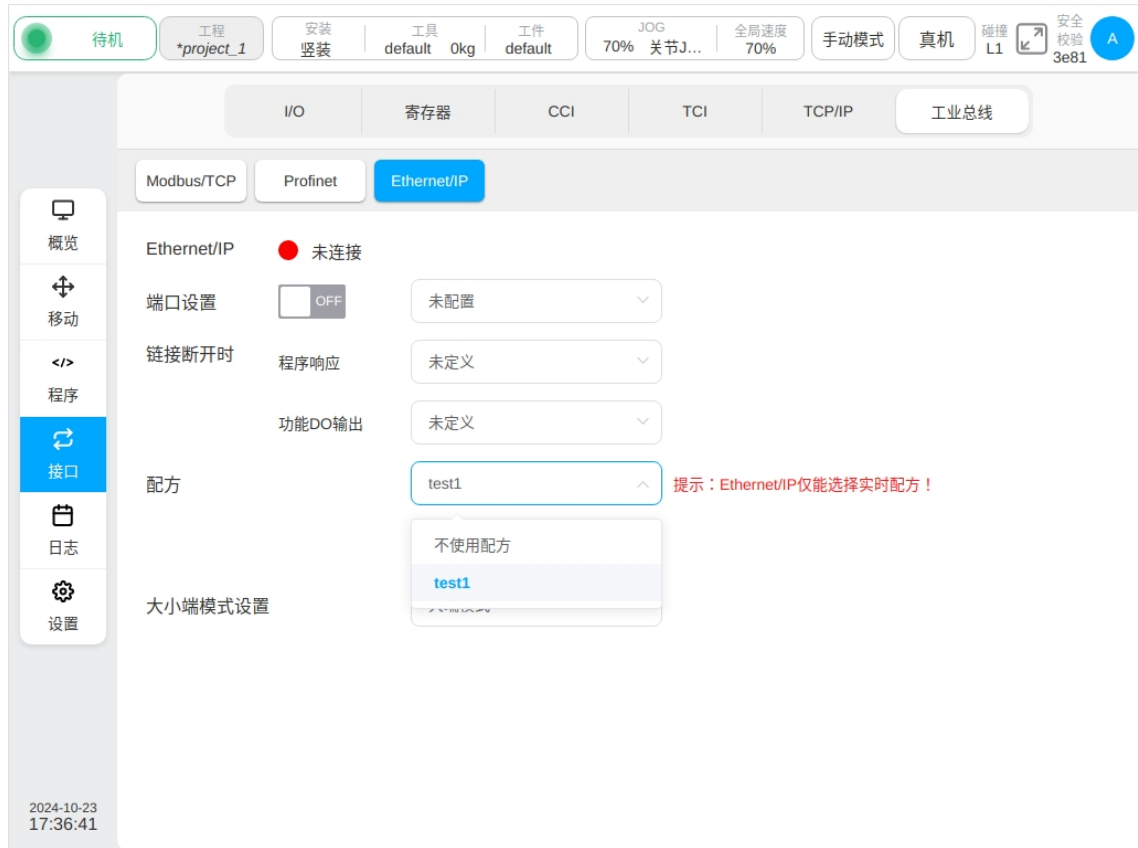
配方类型: 实时配方

输入数据	输出数据
帧头	帧头
帧尾	帧尾
输入数据总长度(含帧头帧尾)	输出数据总长度(含帧头帧尾)
9	20
__DigitalOutputCommand_1 byte 0	__DigitalSignalStatus_1 byte 0
__BitInputReg_1 byte 1	__DigitalSignalStatus_3 byte 1
__BitInputReg_3 byte 2	__DigitalSignalStatus_4 byte 2
__WordInputReg_1 word 3	__DigitalSignalStatus_7 byte 3
__FloatInputReg_1 float 5	__DigitalSignalStatus_9 byte 4

取消 保存

2024-10-23 10:05:19

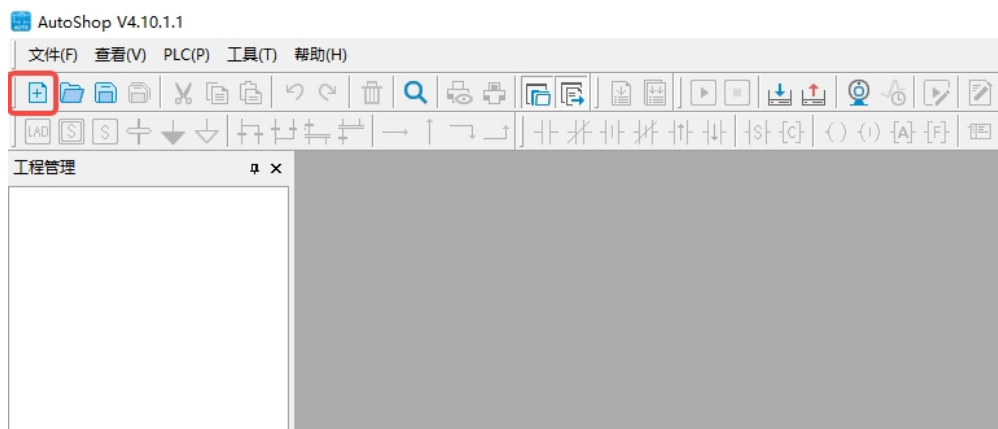
- ・ “ 配方 ” ドロップダウンリストで作成したリアルタイムレシピを選択し、プロジェクトを保存してから再起動し、設定を有効にします。



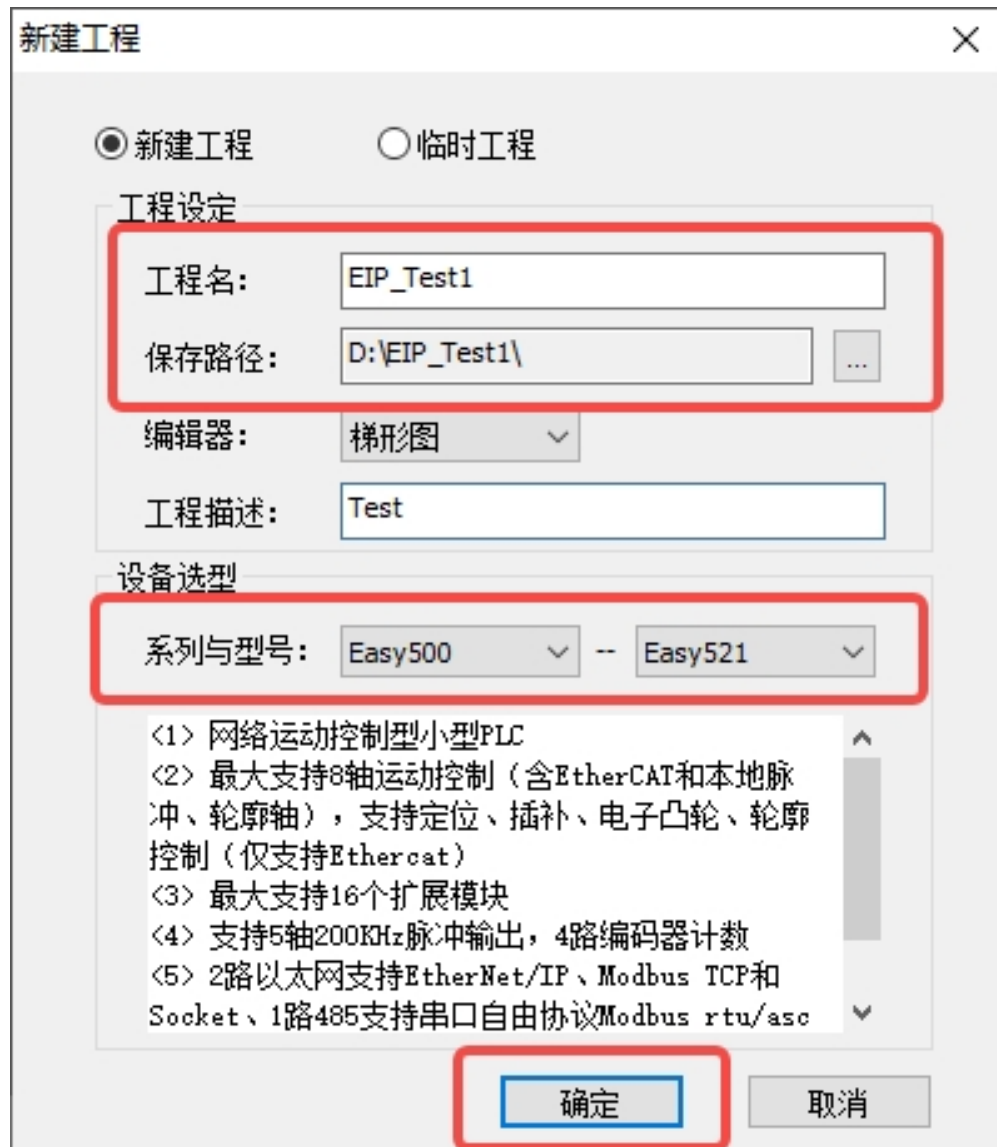
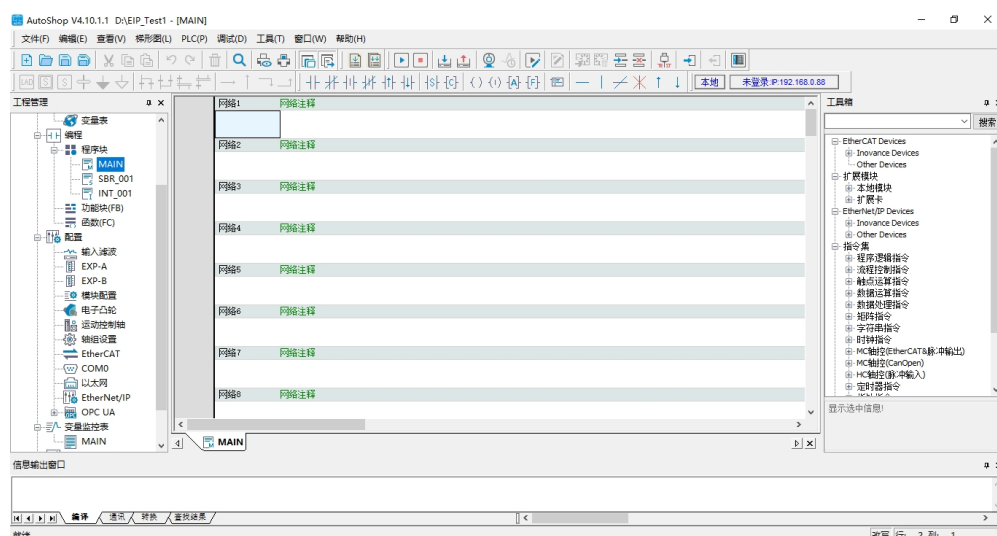
スキャナ（マスター）の設定

以下では、Inovance Easy521 PLC を例に、AutoShop を使用した EtherNet/IP 通信の設定手順を説明します。

- ・ 新規プロジェクトのアイコンをクリックします。

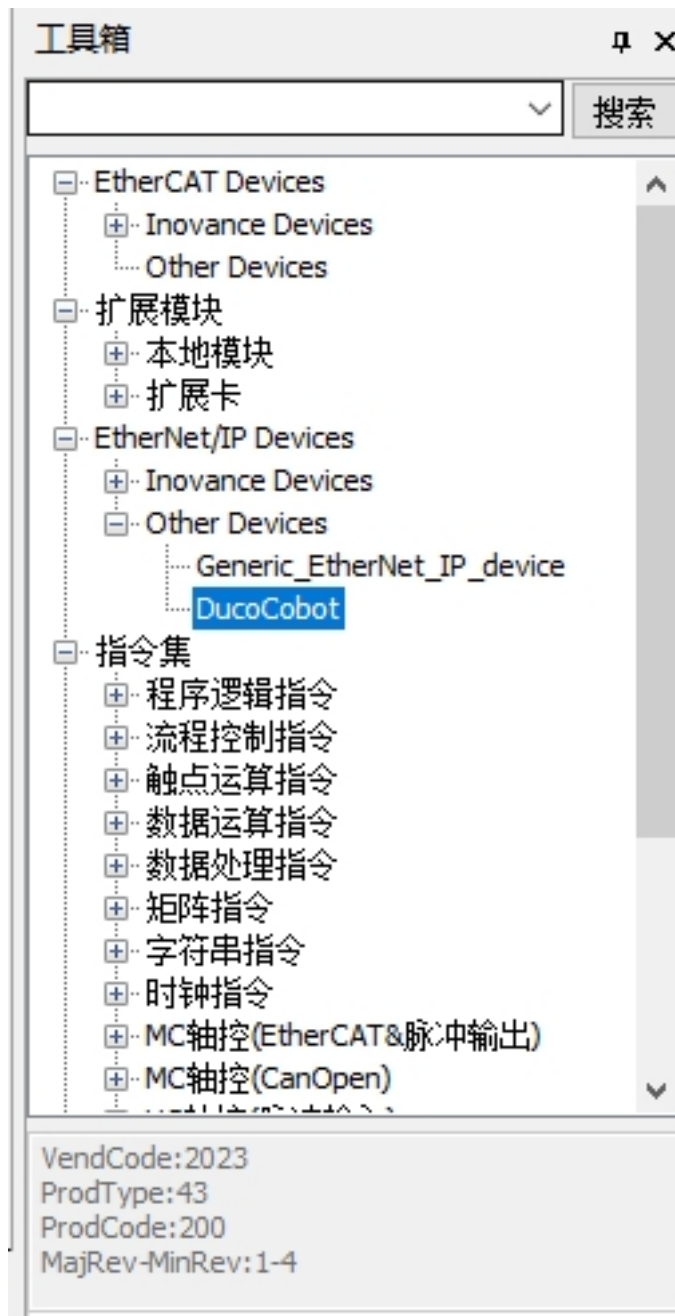


- ・ プロジェクト名と経路（パス）を入力し、対象デバイスのシリーズと型式を選択して、“確定”をクリックします。

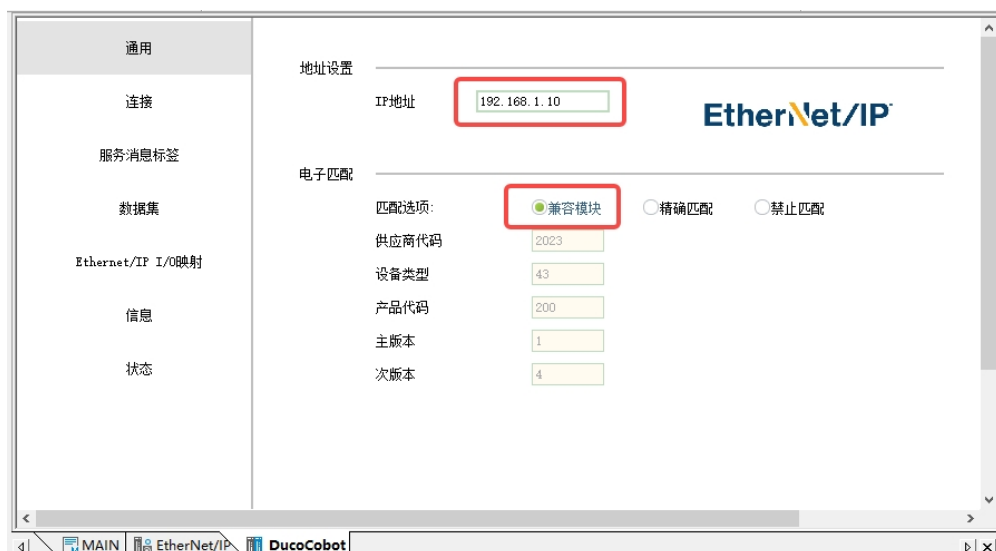


- 右側のツールボックスにある“EtherNet/IP Devices”を右クリックし、表示された“入 EDS”をクリックします。EDS ファイルを選択して“確定”をクリックすると、“Other Devices”リストに“Ducocobot”が表示されます。

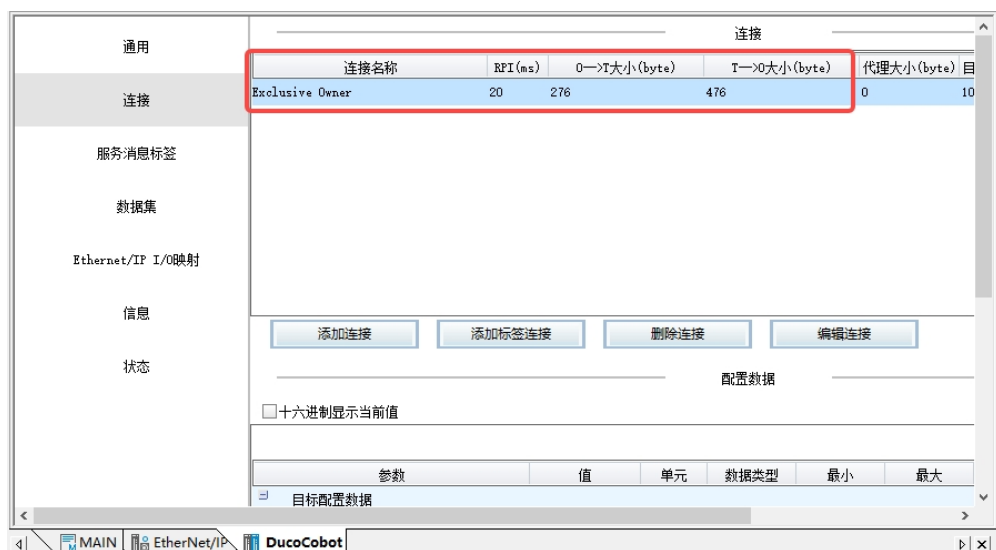




- ・ 左側の“ 工程管理”->“ 配置”->“EtherNet/ IP” をダブルクリックして EtherNet/ IP 設定画面を表示します。右側の“DucoCobot” をダブルクリックし、ロボットスレーブをネットワーク構成に追加します。



- “☒ 接”画面に切り替えます。デフォルトでは“☒ 所有者” (Exclusive Owner) 接続が表示されます。Originator (スキャナ) から Target (対象デバイス) へのデータ量 (O→T size) はデフォルトで 276 byte、Target から Originator へのデータ量 (T→O size) は 476 byte です。要求パケット間隔 (RPI) のデフォルト値は 20 ms で、設定範囲は [5 ms, 1000 ms] です。



- “☒☒☒ 接” をクリックして接続設定画面を開き、入力サイズ、出力サイズ、および RPI を変更します。入出力のバイト数はロボット側の設定と一致させる必要があります。一致しない場合は接続できません。

编辑连接

通用参数

连接路径: 20 04 24 97 2C 96 2C 64

触发类型: 循环的 RPI (ms): 20

传输类型: 专有所有者 超时倍增: 16 *RPI

扫描到目标 (输出)

O->T 大小 (Bytes): 276

代理配置大小 (Bytes): 0

目标配置大小 (Bytes): 10

连接类型: 点对点

连接优先级: Scheduled

固定/变量: 固定

转换格式: 纯数据

禁止时间 (ms): 0

从目标到扫描 (输入)

T->O 大小 (Bytes): 476

连接类型: 点对点

连接优先级: Scheduled

固定/变量: 固定

转换格式: 纯数据

禁止时间 (ms): 0

确定 取消

- ・ ロボット画面でレシピを使用しない場合は、PLC 側の入出力バイト数をデフォルト値に設定します。入出力データ構造については 5.2 章を参照してください。
- ・ 通信データ構造をカスタマイズする場合は、接続設定画面で入出力のバイト数を変更します。
- ・ たとえば入力サイズを 20 byte、出力サイズを 9 byte に設定し、“ 確定 ” をクリックします。ロボット画面のレシピでは、出力サイズを 20 byte、入力サイズを 9 byte に設定します。

编辑连接

通用参数

连接路径: 20 04 24 97 2C 96 2C 64

触发类型: 循环的 RPI (ms): 20

传输类型: 专有所有者 超时倍增: 16 *RPI

扫描到目标 (输出)

O->T 大小 (Bytes): 9

代理配置大小 (Bytes): 0

目标配置大小 (Bytes): 10

连接类型: 点对点

连接优先级: Scheduled

固定/变量: 固定

转换格式: 纯数据

禁止时间 (ms): 0

从目标到扫描 (输入)

T->O 大小 (Bytes): 20

连接类型: 点对点

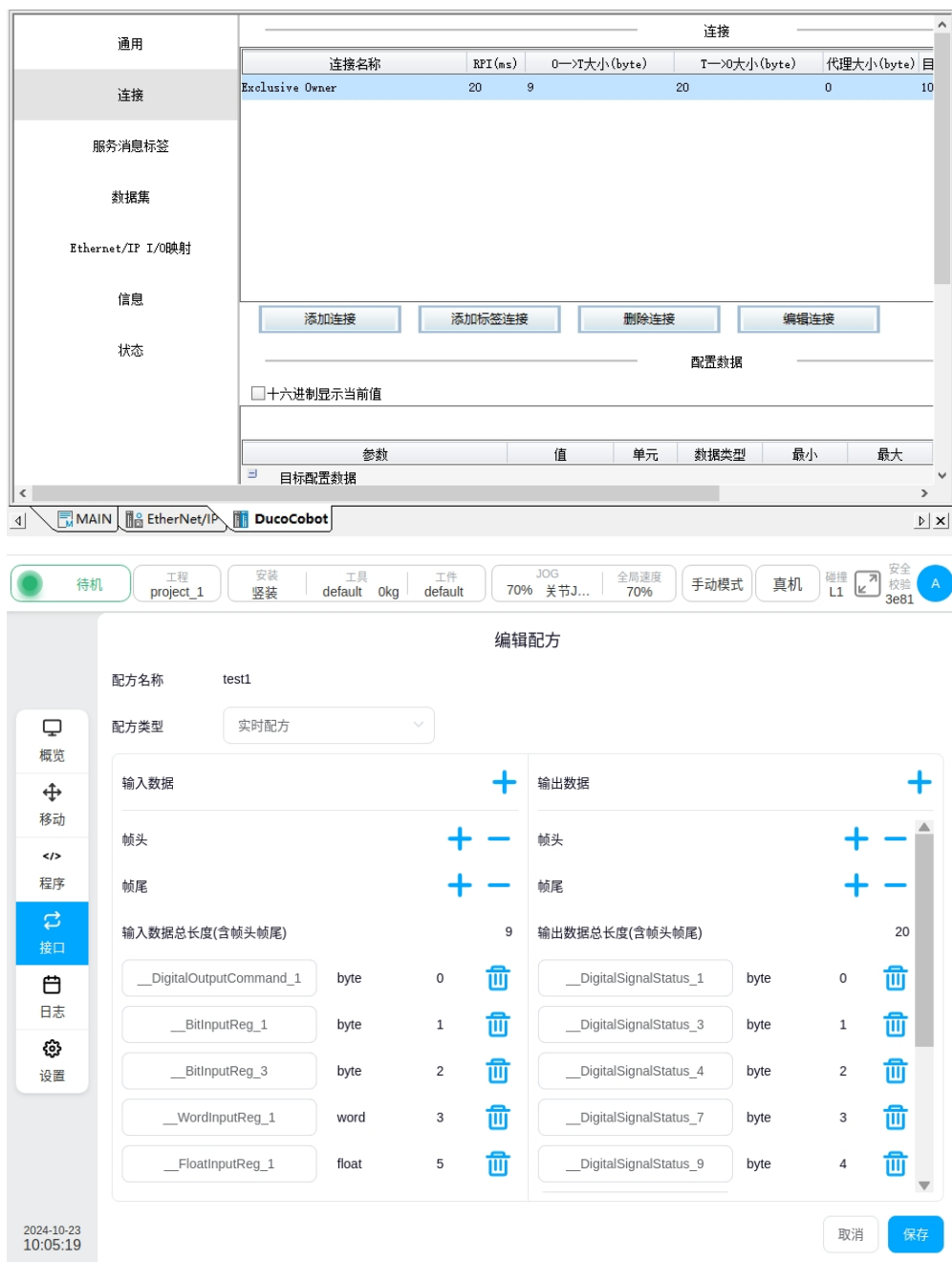
连接优先级: Scheduled

固定/变量: 固定

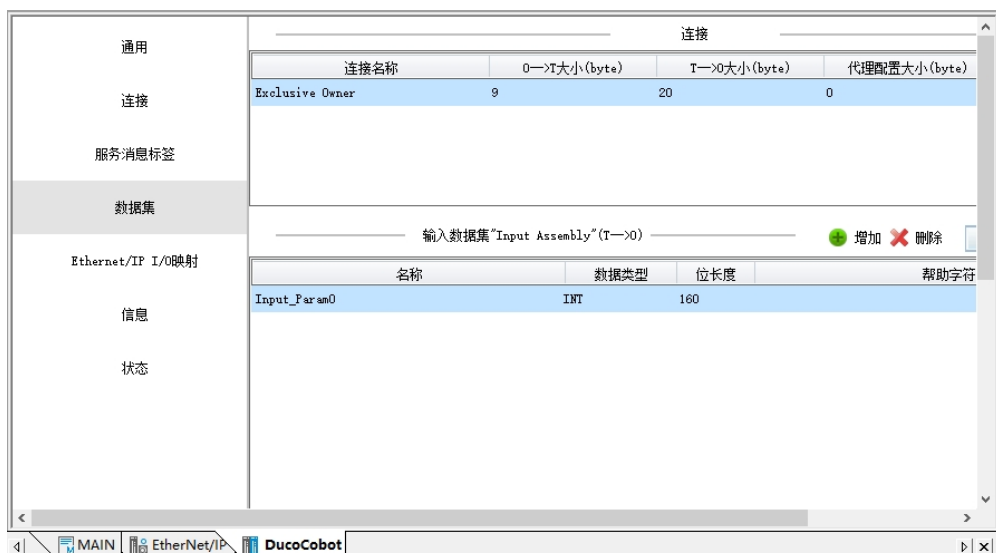
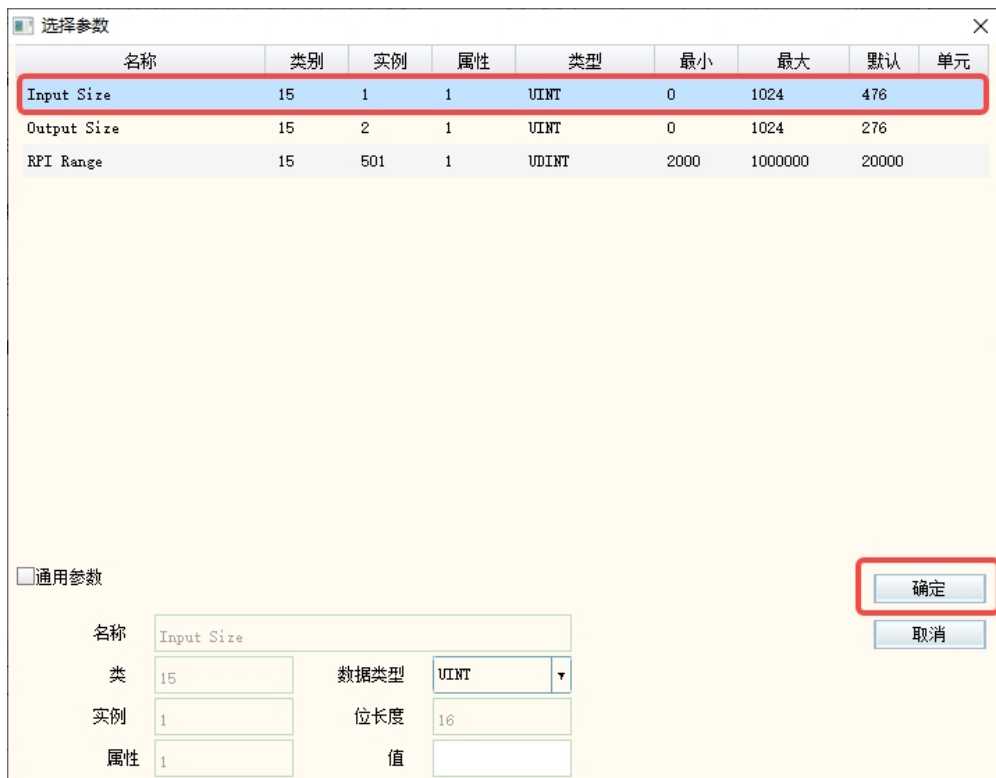
转换格式: 纯数据

禁止时间 (ms): 0

确定 取消

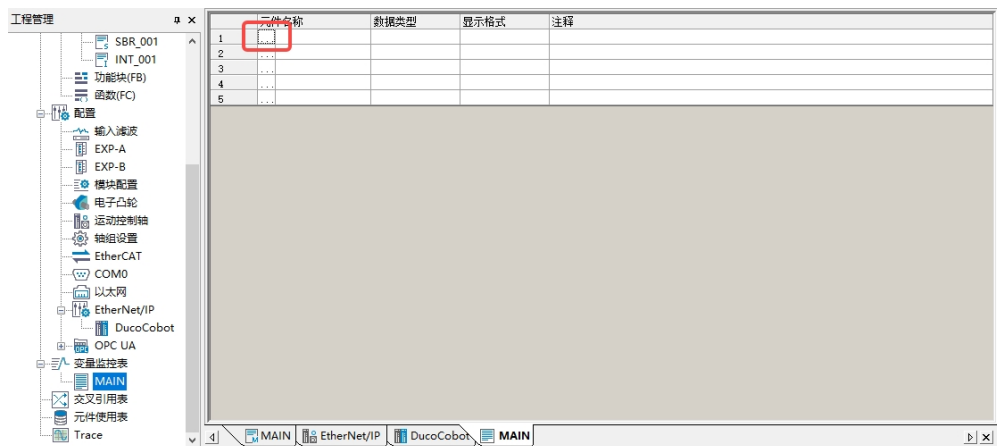


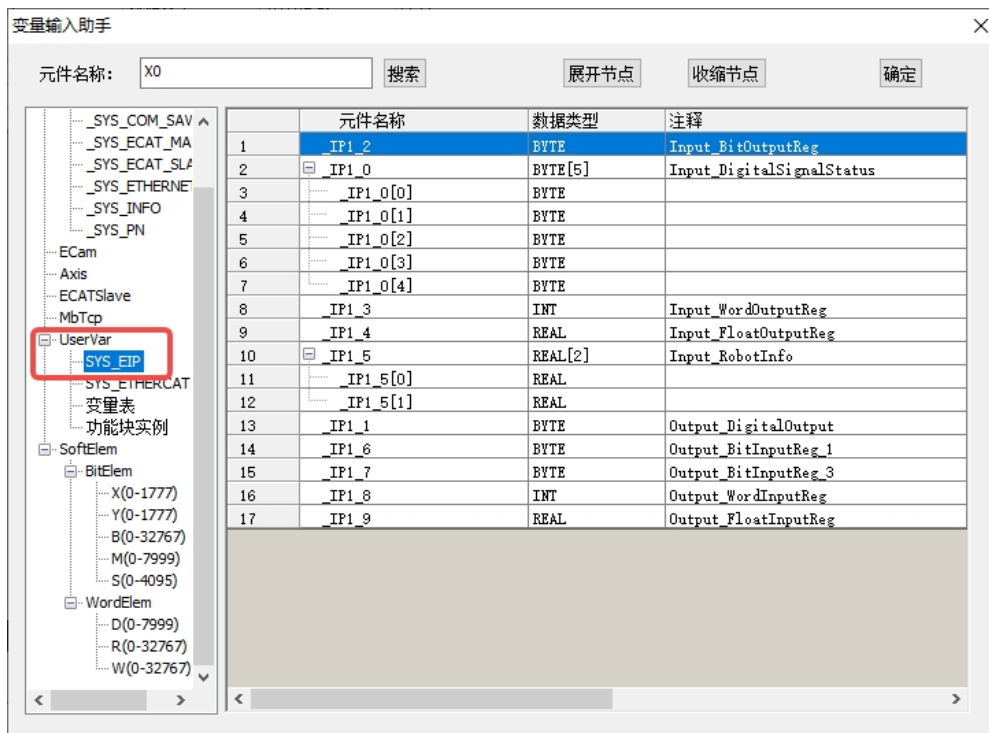
- ・ “数据集” をクリックし、入力データセットと出力データセットを設定します。



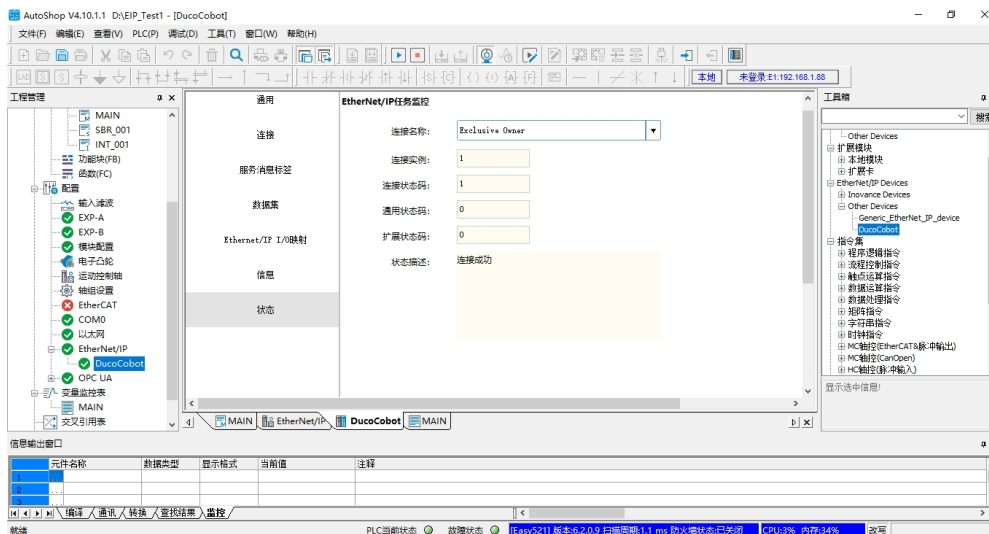


- ・ “工程管理”→“量控表”→“MAIN” をダブルクリックしてデータ監視画面に切り替え、“...” をクリックして作成したデータセットを選択します。





- “下☒”、“运行”、“☒控”の順にクリックします。PLC への接続に成功すると、入出力データを監視できます。





5.2 データ構造

5.2.1 データ構造

ロボットの EtherNet/IP 通信でレシピを使用しない場合、デフォルトでは入力 276 byte、出力 476 byte (64 byte は予約領域、実データは 412 byte) で送受信します。入出力データ構造を以下に示します。

データ方向：ロボット → PLC

(1) R2PLC_DigitalSignalStatus (10 byte)：ロボットデジタル信号状態

データ		
BYTE1	general_digital_input[0..7]	DIN[0]
BYTE2	general_digital_input[8..15]	DIN[1]
BYTE3	tool_digital_input[0..7]	DIN[2]
BYTE4	general_digital_output[0..7]	DIN[3]
BYTE5	general_digital_output[8..15]	DIN[4]
BYTE6	tool_digital_output[0..7]	DIN[5]
BYTE7	fun_digital_input[0..7]	DIN[6]
BYTE8	fun_digital_output[0..7]	DIN[7]
BYTE9	safety_state_input	DIN[8]
BYTE10	safety_state_output	DIN[9]

注：上記の例ではグローバル変数 **DIN** を使用しています。

端子の定義については、『DUCO CORE ハードウェアマニュアル』を参照してください。

general_digital_input: 制御盤汎用 DI 状態。

tool_digital_input: ロボットアーム末端 DI 状態

general_digital_output: 制御盤汎用 DO 状態

tool_digital_output: ロボットアーム末端 DO 状態

fun_digital_input: 制御盤機能 DI 状態

fun_digital_output: 制御盤機能 DO 状態

安全コントローラ状態 safety_state の構造

	safety_state_1	safety_state_2
bit0	ystem_emergency_stop	config_safety_output0
bit1	external_emergency_stop	config_safety_output1
bit2	protective_stop_input	reserved
bit3	operation_mode_input	reserved
bit4	3_position_enable_input	reserved
bit5	config_safety_input0	reserved
bit6	config_safety_input1	reserved
bit7	reserved	reserved

config_safety_input: 画面で設定した安全入力

config_safety_output: 画面で設定した安全出力

(2) R2PLC_Reserved (6 byte) : 予約

BYTE1..6 Reserved DIN[10...15]

(3) R2PLC_RobotState (32float) : ロボット状態情報出力

Float1..7	joint_pos (rad)	DIN[16...43]
Float8..13	tcp_pose (TCP is relative to the base coordinates)	DIN[44...67]
Float14..19	tcp_force	DIN[68...91]
Float20..25	tcp_offset	DIN[92...115]
Float26..29	tcp_load (centroid, mass)	DIN[116...131]
Float30..32	reserved	DIN[132...143]

注: 上記の例ではグローバル変数 DIN を使用しています。

(4) R2PLC_BitOutputReg (10 byte) : ビット出力レジスタ。ロボットの現在のビットレジスタ状態を出力

BYTE1	fun_registers_output[1..8]	DIN[144]
BYTE2	fun_registers_output[9..16]	DIN[145]
BYTE3	bool_registers_output[1..8]	DIN[146]
BYTE4	bool_registers_output[9..16]	DIN[147]
BYTE5	bool_registers_output[17..24]	DIN[148]
BYTE6	bool_registers_output[25..32]	DIN[149]
BYTE7	bool_registers_output[33..40]	DIN[150]
BYTE8	bool_registers_output[41..48]	DIN[151]
BYTE9	bool_registers_output[49..56]	DIN[152]
BYTE10	bool_registers_output[57..64]	DIN[153]

注：上記の例ではグローバル変数 **DIN** を使用しています。

(5) R2PLC_WordOutputReg (64 byte) : Word 出力レジスタ

Word1..32	word_output_register [1..32]	DIN[154...217]
-----------	------------------------------	----------------

(2) R2PLC_Reserved (2 byte) : 予約

BYTE1..2	Reserved	DIN[218...219]
----------	----------	----------------

注：上記の例ではグローバル変数 **DIN** を使用しています。

(6) R2PLC_FloatOutputReg (32 float) : 浮動小数点出力レジスタ

Float1..32	float_output_register [1..32]	DIN[220...347]
------------	-------------------------------	----------------

注：上記の例ではグローバル変数 **DIN** を使用しています。

(7) R2PLC_RobotInfo (16 float) : TCP 合成並進速度など

Float1	Global speed percentage	DIN[348...351]
Float2	Jog speed percentage	DIN[352...355]
Float3	End merge speed	DIN[356...359]
Float4	Real/Emulation mode	DIN[360...363]
Float5	Error code (last error code)	DIN[364...367]
Float6..16	Reserved	DIN[368...411]

注：上記の例ではグローバル変数 **DIN** を使用しています。

データ方向：PLC→ ロボット

(8) PLC2R_Digital_Output_Command (3 byte) : ロボット制御コマンド入力

BYTE1	general_digital_ouput[1..8]	DOUT[0]
BYTE2	general_digital_ouput[9..16]	DOUT[1]
BYTE3	tool_digital_output[1..8]	DOUT[2]

注：上記の例ではグローバル変数 DOUT を使用しています。

(9) PLC2R_Reserved (7 byte) : 予約

BYTE1..7	Reserved	DOUT[3..9]
----------	----------	------------

注：上記の例ではグローバル変数 DOUT を使用しています。

(10) BitInputReg (10 byte) : 汎用ビット入力レジスタ

BYTE1	fun_input_register[1..8]	DOUT[10]
BYTE2	fun_input_register[9..16]	DOUT[11]
BYTE3..10	bit_input_register[1..64]	DOUT[12...19]

注：上記の例ではグローバル変数 DOUT を使用しています。

(11) PLC2R_WordInputReg (64 byte) : Word 入力レジスタ

BYTE1..32	word_input_register[1..32]	DOUT[20...83]
-----------	----------------------------	---------------

注：上記の例ではグローバル変数 DOUT を使用しています。

(12) PLC2R_FloatInputReg (32 float) : 汎用浮動小数点入力レジスタ

Float1..32	float_input_register[1..32]	DOUT[84...211]
------------	-----------------------------	----------------

注：上記の例ではグローバル変数 DOUT を使用しています。

(13) PLC2R_RobotInfo (16 float) : グローバル速度オーバーライドなどの情報を返します。

Float1	Global speed percentage	DOUT[212...215]
Float2..16	Reserved	DOUT[216...275]

注：上記の例ではグローバル変数 DOUT を使用しています。